Data Visualization - 5. Dplyr and Geoms

Kieran Healy

Code Horizons

October 2, 2024

Work with dplyr and ggplot

Load our libraries

```
library(here)      # manage file paths
library(socviz)    # data and some useful functions
library(tidyverse) # your friend and mine
```

Tidyverse components

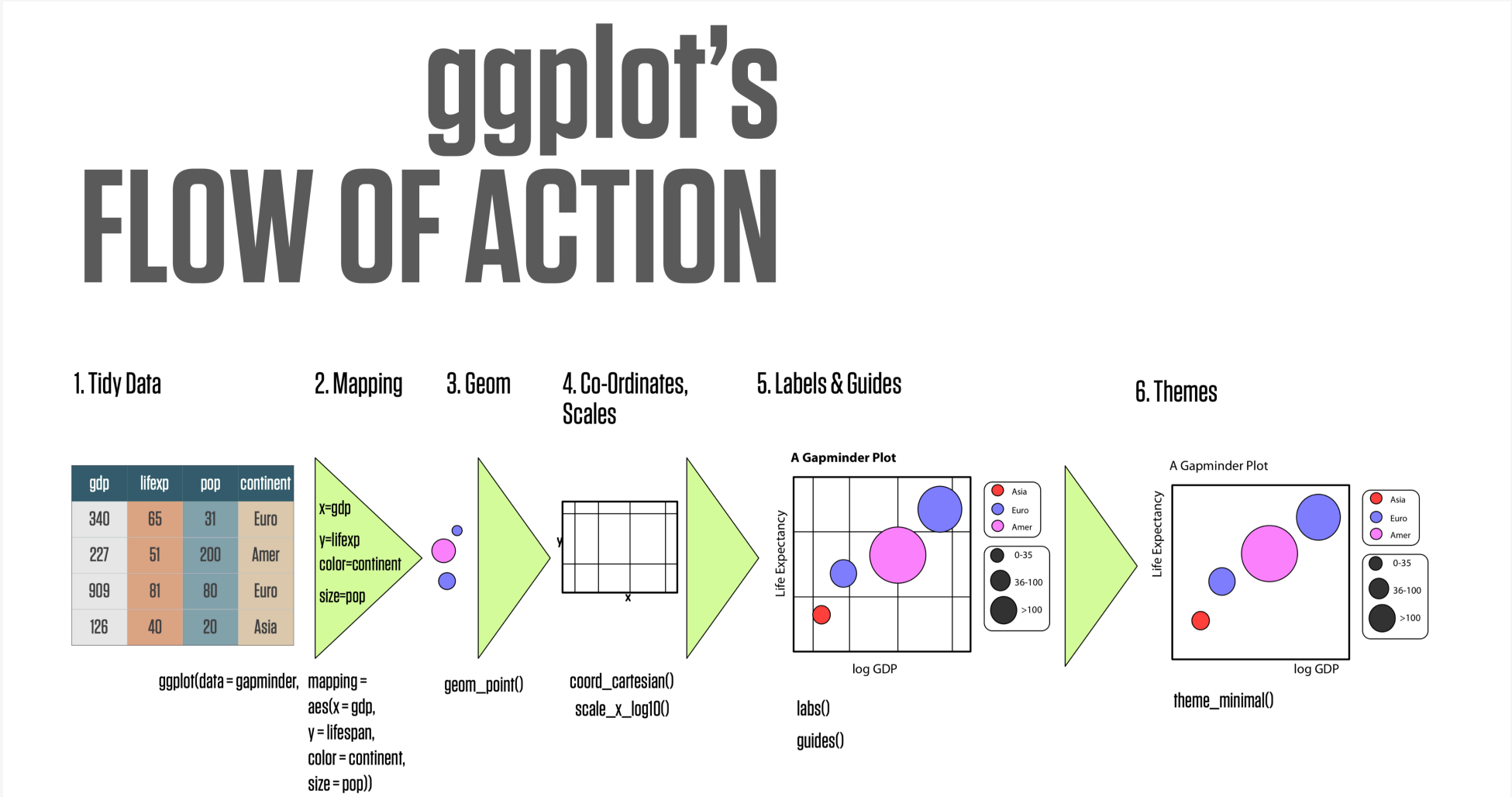
```
library(tidyverse)
Loading tidyverse: ggplot2
Loading tidyverse: tibble
Loading tidyverse: tidyr
Loading tidyverse: readr
Loading tidyverse: purrr
Loading tidyverse: dplyr
```

Load the package and ...

- ◁ **Draw graphs**
- ◁ **Nicer data tables**
- ◁ **Tidy your data**
- ◁ **Get data into R**
- ◁ **Fancy Iteration**
- ◁ **Action verbs for tables**

Other tidyverse components

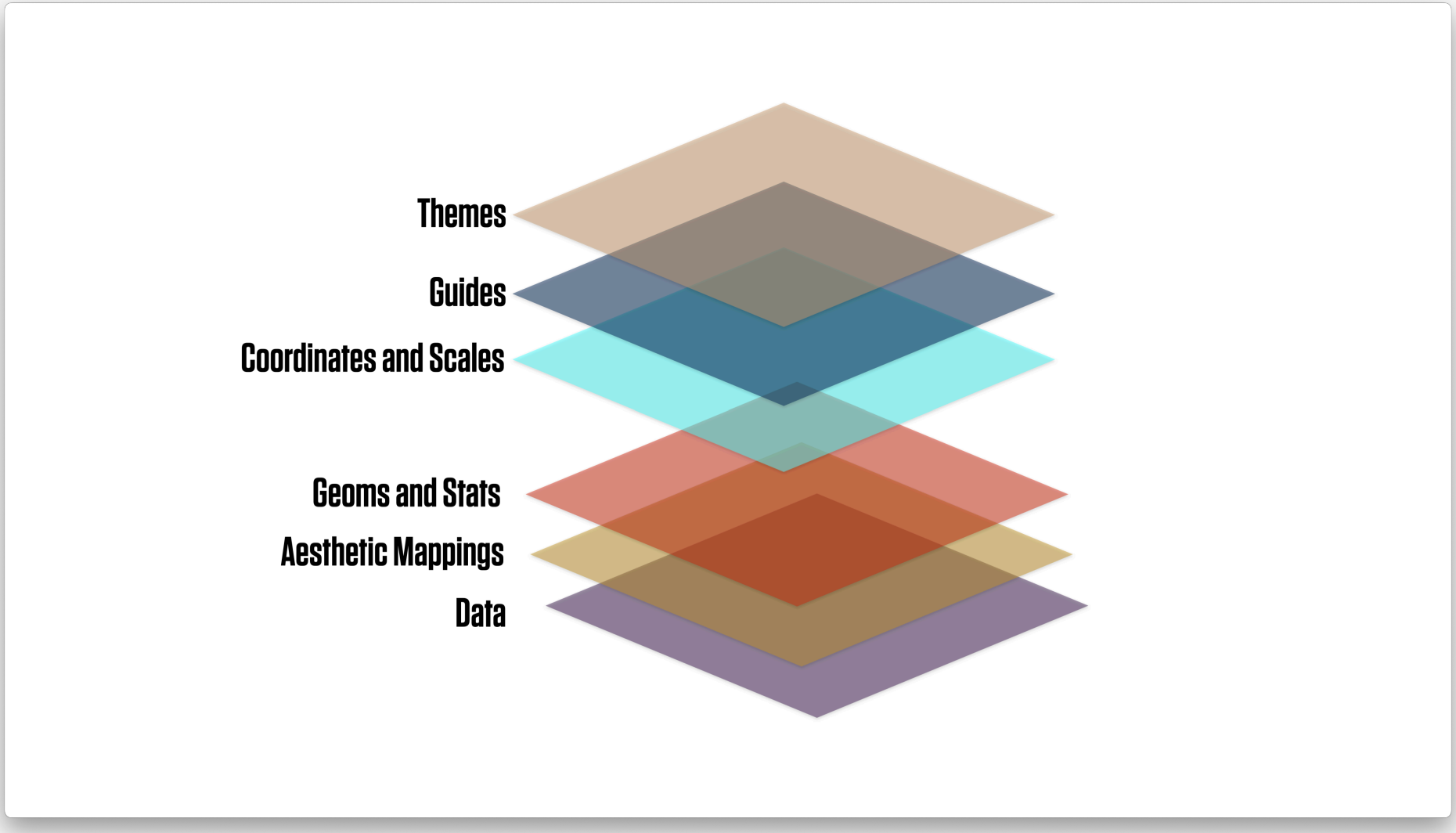
`forcats`	◁ **Deal with factors**
`haven`	◁ **Import Stata, SPSS, etc**
`lubridate`	◁ **Dates, Durations, Times**
`readxl`	◁ **Import from spreadsheets**
`stringr`	◁ **Strings and Regular Expressions**
`reprex`	◁ **Make reproducible examples**



ggplot's flow of action

Thinking in terms of layers

Thinking in terms of layers



Thinking in terms of layers

Feeding data to ggplot

Transform and summarize first. Then send your clean tables to ggplot.

Crosstabulation and beyond

U.S. General Social Survey data:

gss_sm

```
gss_sm
```

```
# A tibble: 2,867 × 32
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

We often want summary tables or graphs of data like this.

Two-way tables: Row percents

bigregion	Protestant	Catholic	Jewish	None	Other	Total
Northeast	32.4	33.3	5.5	23.0	5.7	100.0
Midwest	47.1	24.9	0.4	22.8	4.8	100.0
South	62.4	15.4	1.1	16.3	4.8	100.0
West	37.7	24.6	1.6	28.5	7.6	100.0

Two-way tables: Column percents

bigregion	Protestant	Catholic	Jewish	None	Other
Northeast	11.5	25.0	52.9	18.1	17.6
Midwest	23.7	26.5	5.9	25.4	20.8
South	47.4	24.7	21.6	27.5	31.4
West	17.4	23.9	19.6	29.1	30.2
Total	100.0	100.0	100.0	100.0	100.0

Two-way tables: Full marginals

bigregion	Protestant	Catholic	Jewish	None	Other
Northeast	5.5	5.7	0.9	3.9	1.0
Midwest	11.4	6.0	0.1	5.5	1.2
South	22.8	5.6	0.4	6.0	1.8
West	8.4	5.4	0.4	6.3	1.7

dplyr lets you work with tibbles

dplyr's core verbs

`dplyr` draws on the logic and language of **database queries**

Some actions to take on a single table

For each action there's a function

Group and Summarize

General Social Survey data: gss_sm

```
## library(socviz) # if not loaded
gss_sm
```

```
# A tibble: 2,867 × 32
    year    id ballot       age childs sibs   degree race  sex    region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct>  <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male   New E… $170000…
 2  2016     2 2             61      0 3      High … White Male   New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male   New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male   New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male   Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male   Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Summarizing a Table

Here's what we're going to do:

1. Individual-Level GSS Data on Region and Religion

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

2. 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

3. Percent Religious Preferences by Census Region

bigregion	religion	N	pct
Northeast	Protestant	123	28.3
Northeast	Catholic	149	34.3
Northeast	Jewish	15	3.4
Northeast	None	97	22.3
Northeast	Christian	14	3.2
Northeast	Other	31	7.1

Summarizing a Table

```
gss_sm ▷
  select(id, bigregion, religion)
```

```
# A tibble: 2,867 × 3
      id bigregion religion
   <dbl> <fct>     <fct>
 1     1 Northeast None
 2     2 Northeast None
 3     3 Northeast Catholic
 4     4 Northeast Catholic
 5     5 Northeast None
 6     6 Northeast None
 7     7 Northeast None
 8     8 Northeast Catholic
 9     9 Northeast Protestant
10    10 Northeast None
# ℹ 2,857 more rows
```

We're just taking a look at the relevant columns here.

Group by *one* column or variable

```
gss_sm ▷
  group_by(bigregion)
```

```
# A tibble: 2,867 × 32
# Groups:   bigregion [4]
    year    id ballot       age childs sibs   degree race  sex    region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct>  <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male   New E… $170000…
 2  2016     2 2             61      0 3      High … White Male   New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male   New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema…  New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema…  New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema…  New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male   New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema…  Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male   Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male   Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
```

Grouping just changes the logical structure of the tibble.

Group and summarize by *one* column

```
gss_sm
```

```
# A tibble: 2,867 × 32
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *one* column

```
gss_sm ▷
  group_by(bigregion)
```

```
# A tibble: 2,867 × 32
# Groups:   bigregion [4]
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *one* column

```
gss_sm ▷
  group_by(bigregion) ▷
  summarize(total = n())
```

```
# A tibble: 4 × 2
  bigregion total
  <fct>     <int>
1 Northeast   488
2 Midwest     695
3 South      1052
4 West        632
```

Group and summarize by *two* columns

```
gss_sm
```

```
# A tibble: 2,867 × 32
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *two* columns

```
gss_sm ▷
  group_by(bigregion, religion)
```

```
# A tibble: 2,867 × 32
# Groups:   bigregion, religion [24]
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *two* columns

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n())
```

```
# A tibble: 24 × 3
# Groups:   bigregion [4]
   bigregion religion   total
   <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
# ℹ 14 more rows
```

Calculate frequencies

```
gss_sm
```

```
# A tibble: 2,867 × 32
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Calculate frequencies

```
gss_sm ▷
  group_by(bigregion, religion)
```

```
# A tibble: 2,867 × 32
# Groups:   bigregion, religion [24]
    year    id ballot       age childs sibs   degree race  sex   region income16
   <dbl> <dbl> <labelled> <dbl>  <dbl> <labe> <fct>  <fct> <fct> <fct>  <fct>
 1  2016     1 1             47      3 2      Bache… White Male  New E… $170000…
 2  2016     2 2             61      0 3      High … White Male  New E… $50000 …
 3  2016     3 3             72      2 3      Bache… White Male  New E… $75000 …
 4  2016     4 1             43      4 3      High … White Fema… New E… $170000…
 5  2016     5 3             55      2 2      Gradu… White Fema… New E… $170000…
 6  2016     6 2             53      2 2      Junio… White Fema… New E… $60000 …
 7  2016     7 1             50      2 2      High … White Male  New E… $170000…
 8  2016     8 3             23      3 6      High … Other Fema… Middl… $30000 …
 9  2016     9 1             45      3 5      High … Black Male  Middl… $60000 …
10  2016    10 3             71      4 1      Junio… White Male  Middl… $60000 …
# ℹ 2,857 more rows
# ℹ 21 more variables: relig <fct>, marital <fct>, padeg <fct>, madeg <fct>,
#   partyid <fct>, polviews <fct>, happy <fct>, partners <fct>, grass <fct>,
#   zodiac <fct>, pres12 <labelled>, wtssall <dbl>, income_rc <fct>,
#   agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>, religion <fct>,
#   bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Calculate frequencies

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n())
```

```
# A tibble: 24 × 3
# Groups:   bigregion [4]
   bigregion religion   total
   <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
# ℹ 14 more rows
```

Calculate frequencies

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n()) ▷
  mutate(freq = total / sum(total),
           pct = round((freq*100), 1))
```

```
# A tibble: 24 × 5
# Groups:   bigregion [4]
   bigregion religion   total    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
# ℹ 14 more rows
```

Pipelines carry assumptions forward

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n()) ▷
  mutate(freq = total / sum(total),
           pct = round((freq*100), 1))
```

```
# A tibble: 24 × 5
# Groups:   bigregion [4]
   bigregion religion   total    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
# ℹ 14 more rows
```

Groups are carried forward till summarized or explicitly ungrouped

Summary calculations are done on the innermost group, which then

Pipelines carry assumptions forward

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n()) ▷
  mutate(freq = total / sum(total),
           pct = round((freq*100), 1))
```

```
# A tibble: 24 × 5
# Groups:   bigregion [4]
   bigregion religion   total    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
# ℹ 14 more rows
```

`mutate()` is quite clever. See how we can immediately use `freq`, even though we are creating it in the same `mutate()` expression.

Convenience functions

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(total = n()) ▷
  mutate(freq = total / sum(total),
           pct = round((freq*100), 1))
```

```
# A tibble: 24 × 5
# Groups:   bigregion [4]
   bigregion religion   total    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
# ℹ 14 more rows
```

We're going to be doing this `group_by()` … `n()` step a lot. Some shorthand for it would be useful.

Three options for counting up rows

Use `n()`

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  summarize(n = n())
```

```
# A tibble: 24 × 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
# ℹ 14 more rows
```

Group it yourself; result is grouped.

Use `tally()`

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  tally()
```

```
# A tibble: 24 × 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
# ℹ 14 more rows
```

More compact; result is grouped.

Use `count()`

```
gss_sm ▷
  count(bigregion, religion)
```

```
# A tibble: 24 × 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
# ℹ 14 more rows
```

One step; result is not grouped.

Pass results on to ... a table

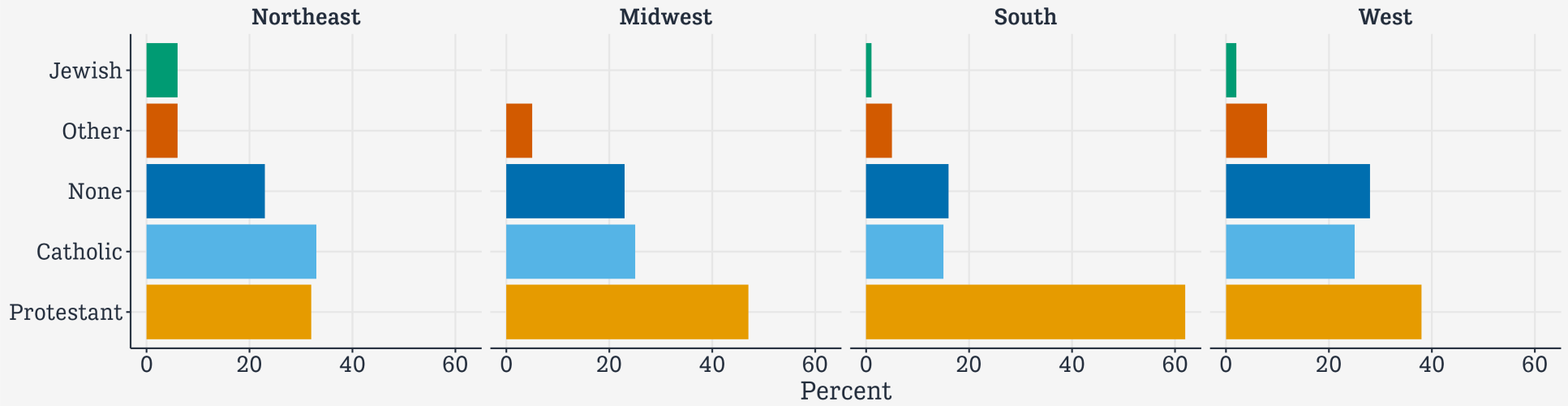
```
gss_sm ▷
  count(bigregion, religion) ▷
  pivot_wider(names_from = bigregion, values_from = n) ▷
  knitr::kable()
```

religion	Northeast	Midwest	South	West
Protestant	158	325	650	238
Catholic	162	172	160	155
Jewish	27	3	11	10
None	112	157	170	180
Other	28	33	50	48
NA	1	5	11	1

More on `pivot_wider()` and `kable()` soon ...

Pass results on to ... a graph

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  tally() ▷
  mutate(pct = round((n/sum(n))*100), 1) ▷
  drop_na() ▷
  ggplot(mapping = aes(x = pct, y = reorder(religion, -pct), fill = religion)) +
  geom_col() +
    labs(x = "Percent", y = NULL) +
    guides(fill = "none") +
    facet_wrap(~ bigregion, nrow = 1)
```



Check by summarizing

```
rel_by_region ← gss_sm ▷
  count(bigregion, religion) ▷
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

```
# A tibble: 24 × 4
   bigregion religion       n   pct
   <fct>     <fct>      <int> <dbl>
 1 Northeast Protestant   158   5.5
 2 Northeast Catholic     162   5.7
 3 Northeast Jewish        27   0.9
 4 Northeast None         112   3.9
 5 Northeast Other         28   1
 6 Northeast <NA>           1   0
 7 Midwest   Protestant   325  11.3
 8 Midwest   Catholic     172   6
 9 Midwest   Jewish         3   0.1
10 Midwest   None         157   5.5
# ℹ 14 more rows
```

Hm, did I sum over right group?

Check by summarizing

```
rel_by_region ← gss_sm ▷
  count(bigregion, religion) ▷
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

```
# A tibble: 24 × 4
   bigregion religion       n   pct
   <fct>     <fct>      <int> <dbl>
 1 Northeast Protestant   158   5.5
 2 Northeast Catholic     162   5.7
 3 Northeast Jewish        27   0.9
 4 Northeast None         112   3.9
 5 Northeast Other         28   1
 6 Northeast <NA>           1   0
 7 Midwest   Protestant   325  11.3
 8 Midwest   Catholic     172   6
 9 Midwest   Jewish         3   0.1
10 Midwest   None         157   5.5
# ℹ 14 more rows
```

Hm, did I sum over right group?

```
## Each region should sum to ~100
rel_by_region ▷
  group_by(bigregion) ▷
  summarize(total = sum(pct))
```

```
# A tibble: 4 × 2
  bigregion total
  <fct>     <dbl>
1 Northeast  17
2 Midwest    24.3
3 South      36.7
4 West       22
```

No! What has gone wrong here?

Check by summarizing

```
rel_by_region ← gss_sm ▷
  count(bigregion, religion) ▷
  mutate(pct = round((n/sum(n))*100, 1))
```

`count()` returns ungrouped results, so there are no groups carry forward to the `mutate()` step.

```
rel_by_region ▷
  summarize(total = sum(pct))
```

```
# A tibble: 1 × 1
  total
  <dbl>
1   100
```

With `count()`, the `pct` values here are the marginals for the whole table.

Check by summarizing

```
rel_by_region ← gss_sm ▷
  count(bigregion, religion) ▷
  mutate(pct = round((n/sum(n))*100, 1))
```

`count()` returns ungrouped results, so there are no groups carry forward to the `mutate()` step.

```
rel_by_region ▷
  summarize(total = sum(pct))
```

```
# A tibble: 1 × 1
  total
  <dbl>
1   100
```

With `count()`, the `pct` values here are the marginals for the whole table.

```
rel_by_region ← gss_sm ▷
  group_by(bigregion, religion) ▷
  tally() ▷
  mutate(pct = round((n/sum(n))*100, 1))
```

```
# Check
rel_by_region ▷
  group_by(bigregion) ▷
  summarize(total = sum(pct))
```

```
# A tibble: 4 × 2
  bigregion total
  <fct>     <dbl>
1 Northeast 100
2 Midwest    99.9
3 South     100
4 West      100.
```

We get some rounding error because we used `round()` after summing originally.

Two lessons

Check your tables!

Two lessons

Inspect your pipes!

Use dplyr to make summary tables.

Then send your clean tables to ggplot.

Facets are often better than Guides

Let's put that table in an object

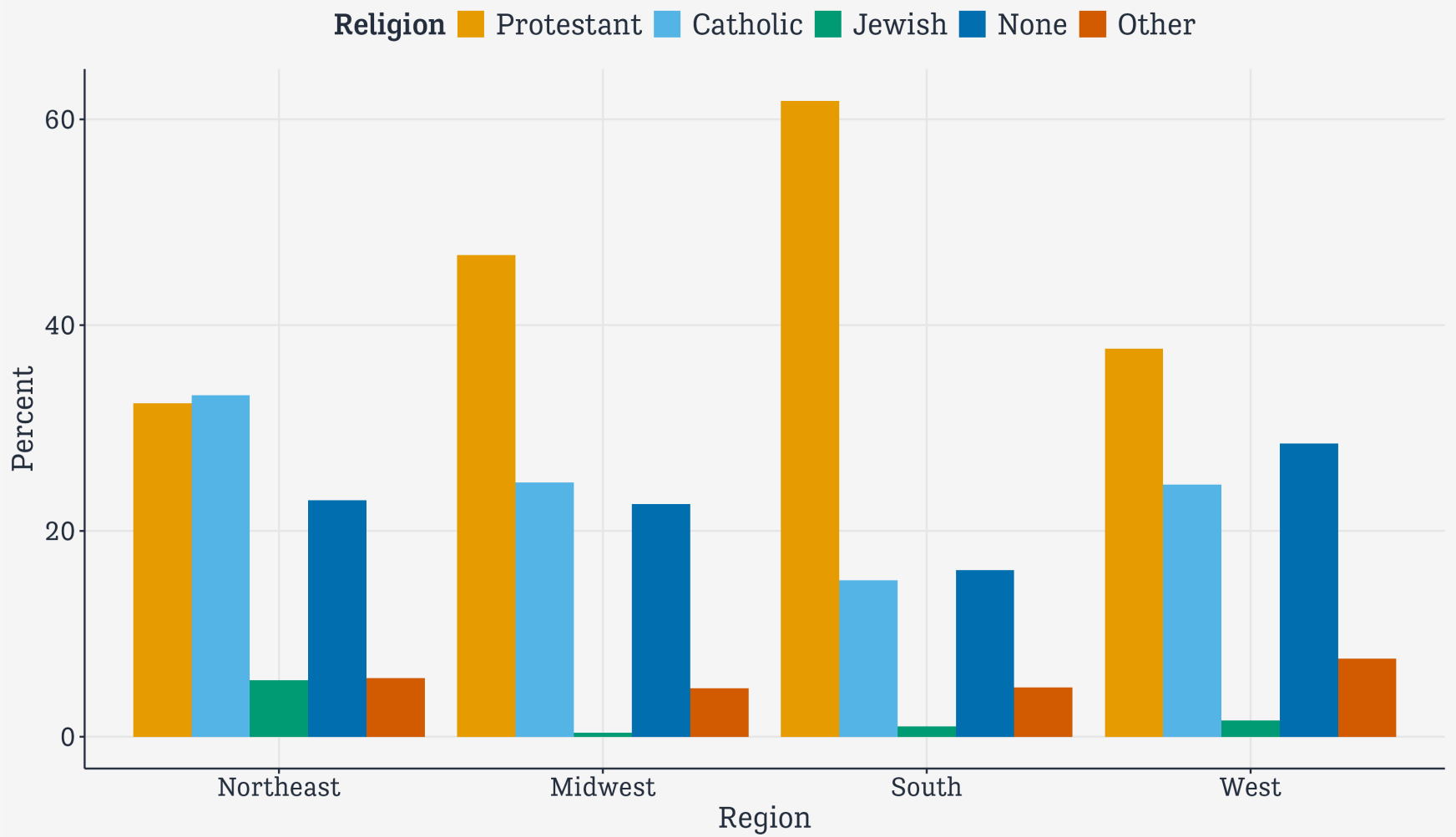
```
rel_by_region ← gss_sm ▷
  group_by(bigregion, religion) ▷
  tally() ▷
  mutate(pct = round((n/sum(n))*100, 1)) ▷
  drop_na()


head(rel_by_region)
```

```
# A tibble: 6 × 4
# Groups:   bigregion [2]
  bigregion religion       n   pct
  <fct>     <fct>      <int> <dbl>
1 Northeast Protestant   158  32.4
2 Northeast Catholic     162  33.2
3 Northeast Jewish        27   5.5
4 Northeast None         112  23
5 Northeast Other         28   5.7
6 Midwest   Protestant   325  46.8
```

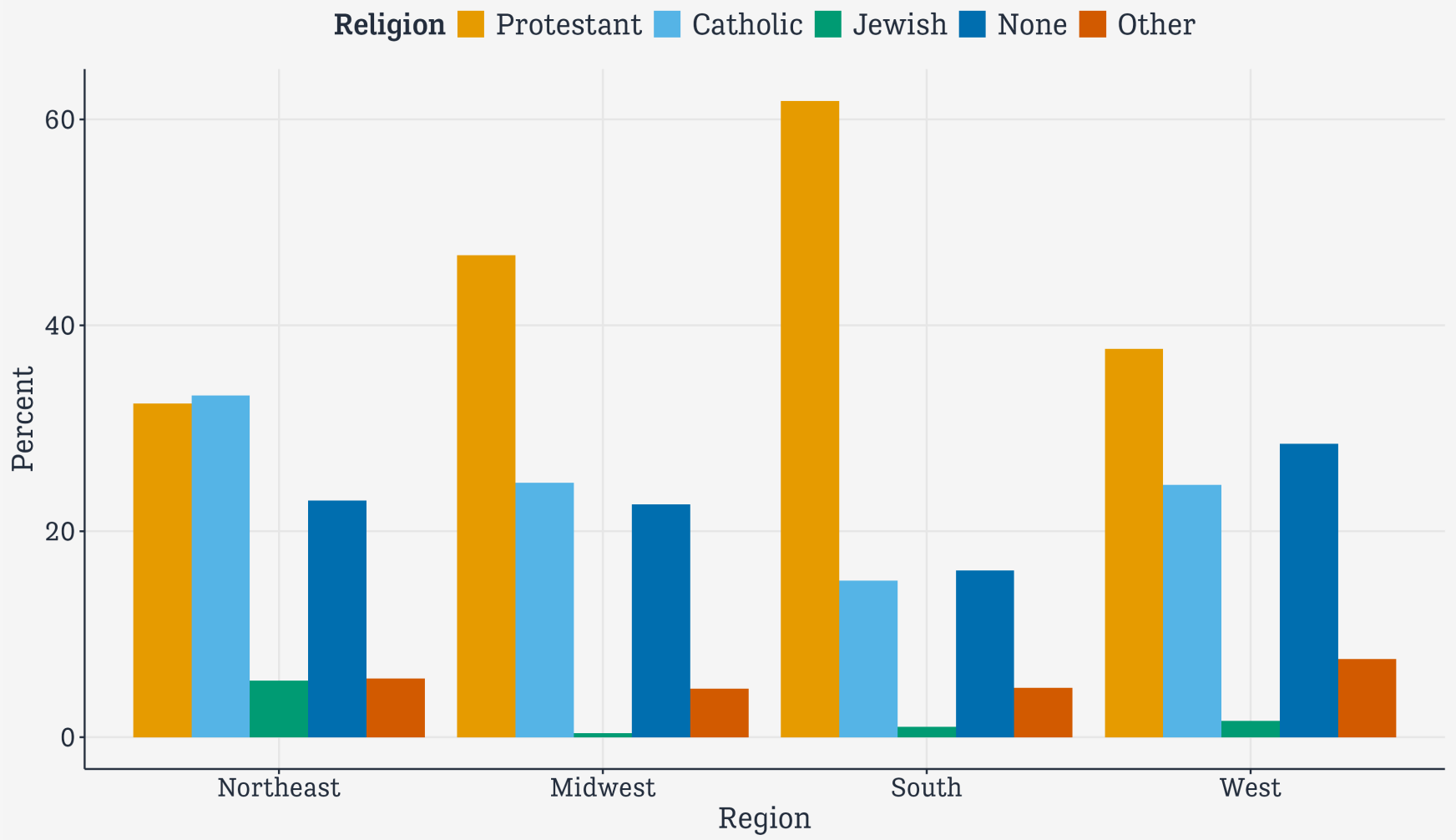
We might write ...

```
p ← ggplot(data = rel_by_region,
               mapping = aes(x = bigregion,
                             y = pct,
                             fill = religion))
p_out ← p + geom_col(position = "dodge") +
    labs(x = "Region",
         y = "Percent",
         fill = "Religion")
```

We might write ...



Is this an effective graph? Not really!



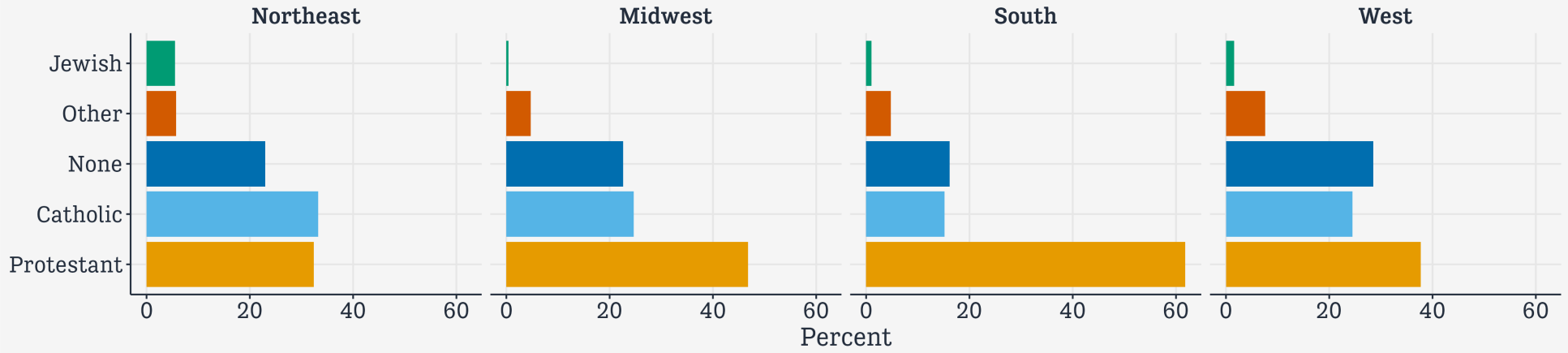
Try faceting instead

```
p ← ggplot(data = rel_by_region,
                mapping = aes(x = pct,
                              y = reorder(religion, -pct),
                              fill = religion))
p_out_facet ← p + geom_col() +
  guides(fill = "none") +
  facet_wrap(~ bigregion, nrow = 1) +
  labs(x = "Percent",
       y = NULL)
```

Putting categories on the y-axis is a very useful trick.

Faceting reduces the number of guides the viewer needs to consult.

Try faceting instead



Try faceting instead

Try putting categories on the y-axis. (And reorder them by x.)

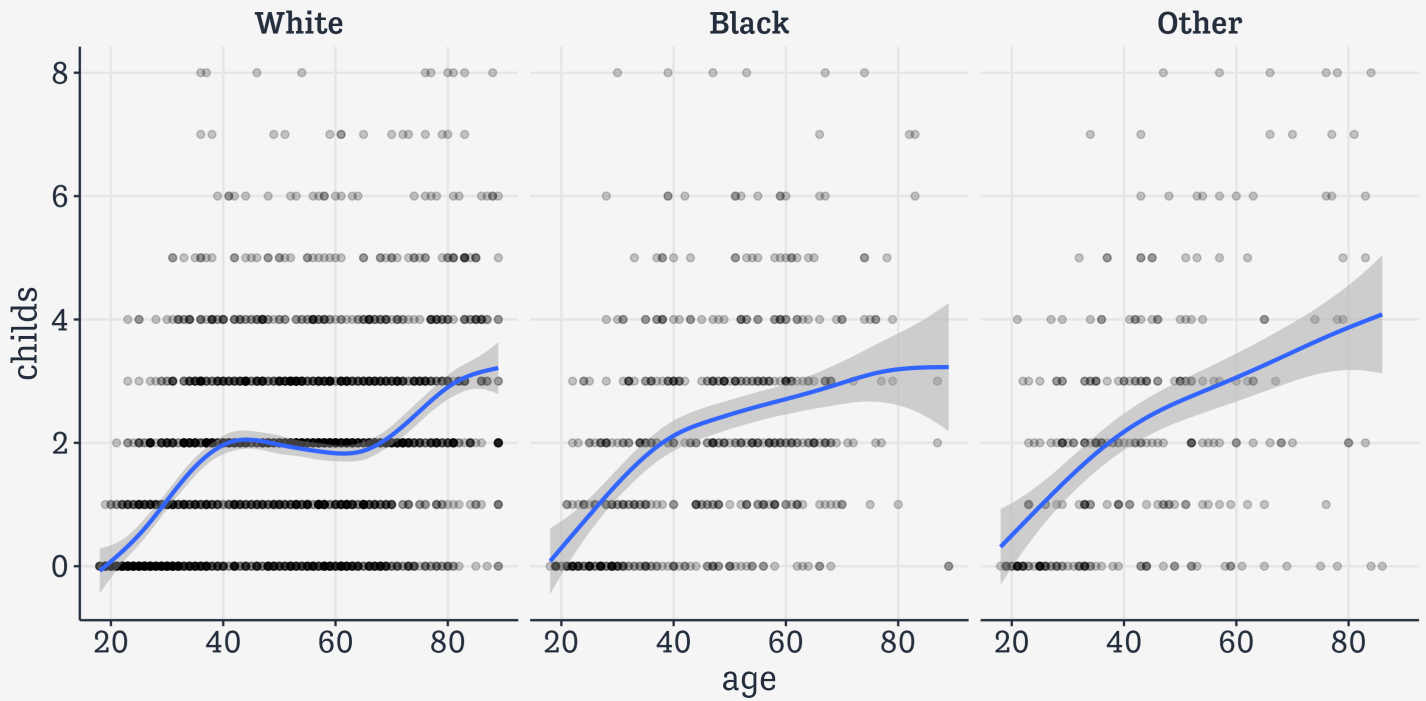
Try faceting variables instead of mapping them to color or shape.

Try to minimize the need for guides and legends.

Two kinds of facet

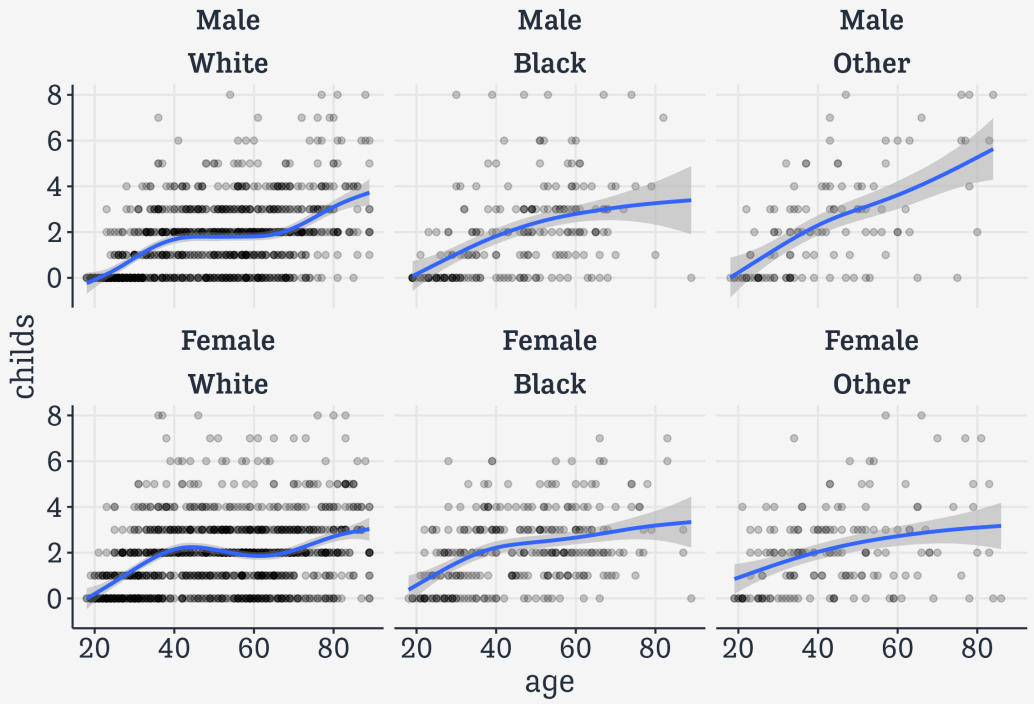
Facet Children vs Age, by Race

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ race)
```



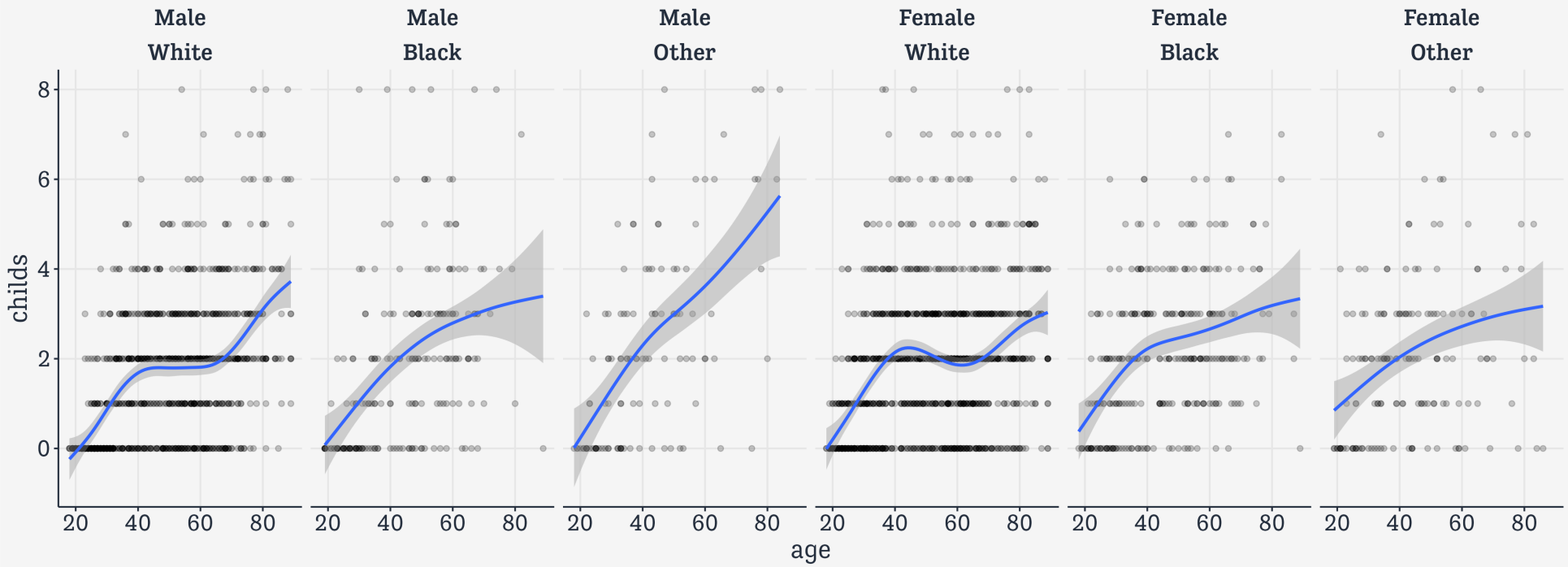
Facet by more than one variable

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ sex + race)
```



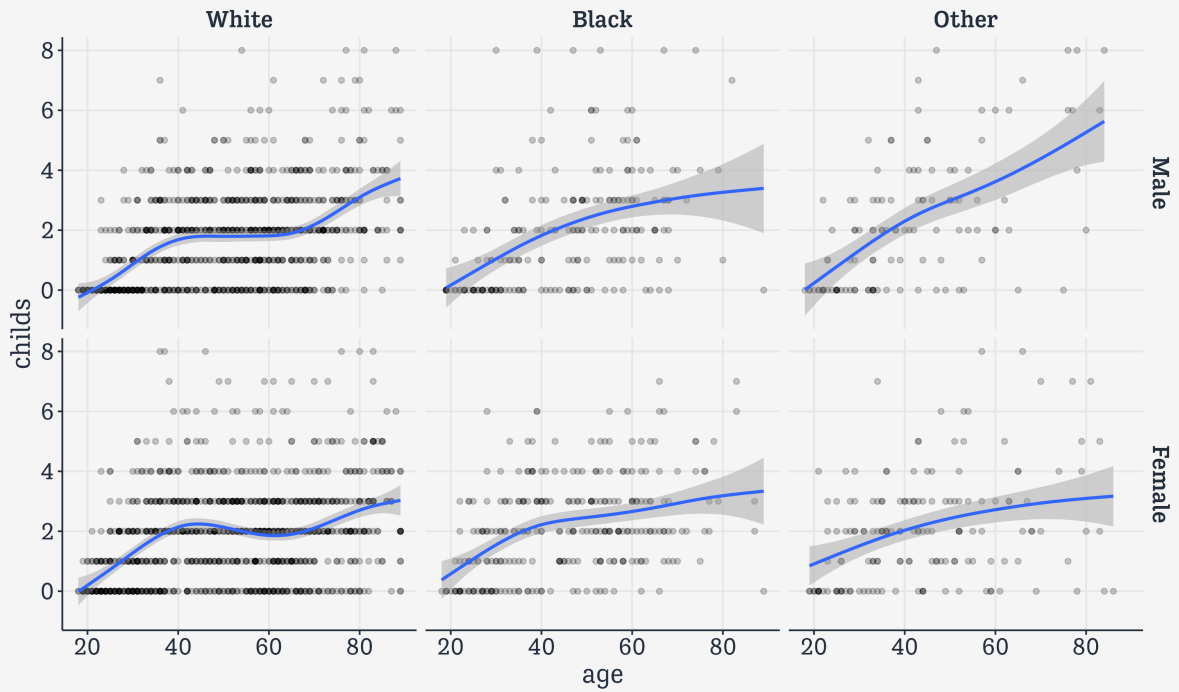
Arrange facet_wrap() quite freely

```
p ←  ggplot(data = gss_sm,
            mapping = aes(x = age, y = childs))

p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_wrap(~ sex + race, nrow = 1)
```



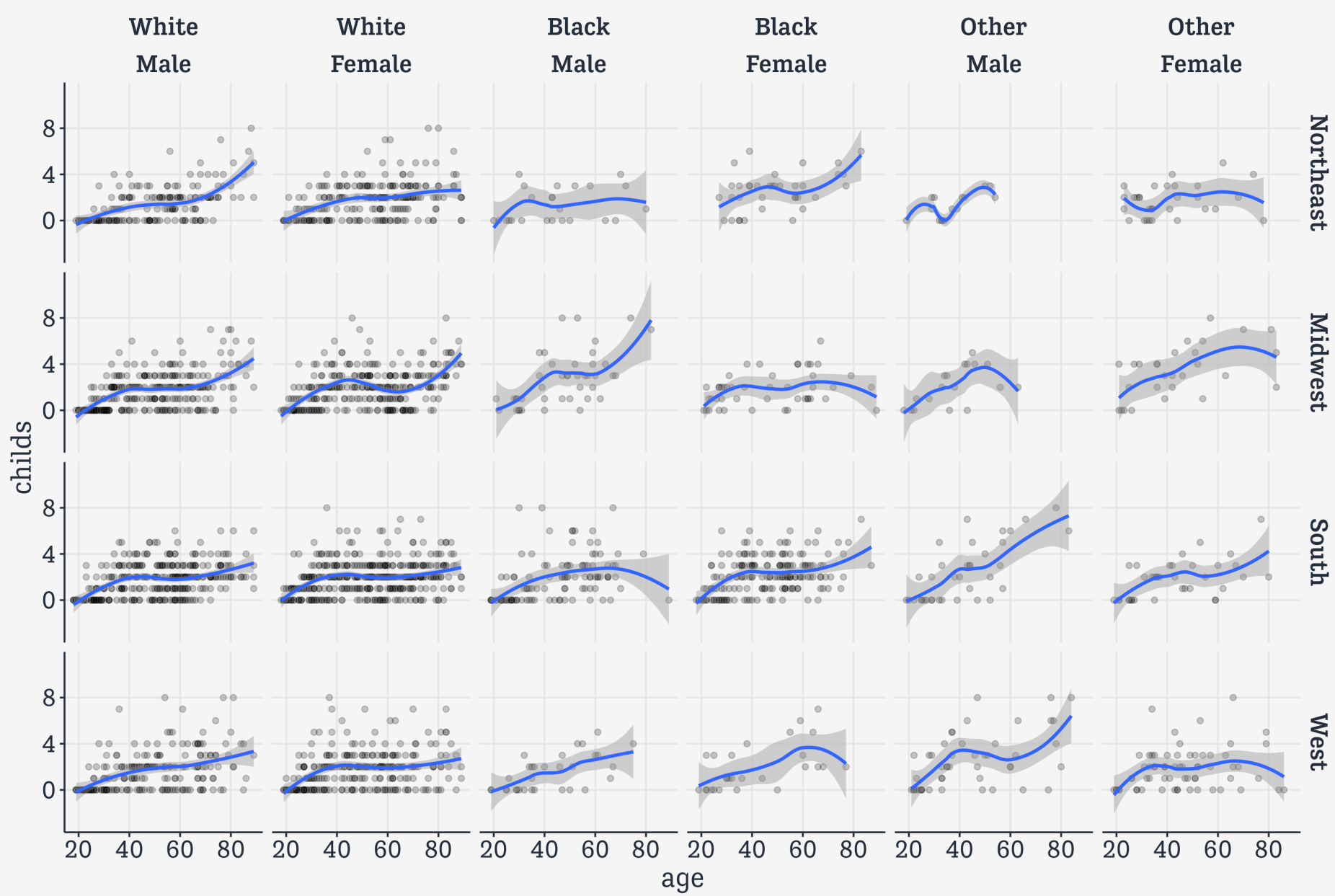
facet_grid() is more like a true crosstab

```
p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_grid(sex ~ race)
```



Extend both to multi-way views

```
p_out ← p + geom_point(alpha = 0.2) +
  geom_smooth() +
  facet_grid(bigregion ~ race + sex)
```


White Male
White Female
Black Male
Black Female
Other Male
Other Female
Northeast
Midwest
South
West
childs
age
0
4
8
20
40
60
80

What we've built-up

Core Grammar

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

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

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

Core grammar

Grouped data; faceting

Along with a few peeks at scale transformations, guide adjustments, and theme adjustment

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

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

All basic steps

dplyr and Pipelining

The elements of filtering and summarizing

```
gss_sm ▷
  group_by(bigregion, religion) ▷
  tally() ▷
  mutate(freq = n / sum(n),
         pct = round((freq*100), 1))
```

```
# A tibble: 24 × 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
# ℹ 14 more rows
```

Example and extension: Organ Donation data

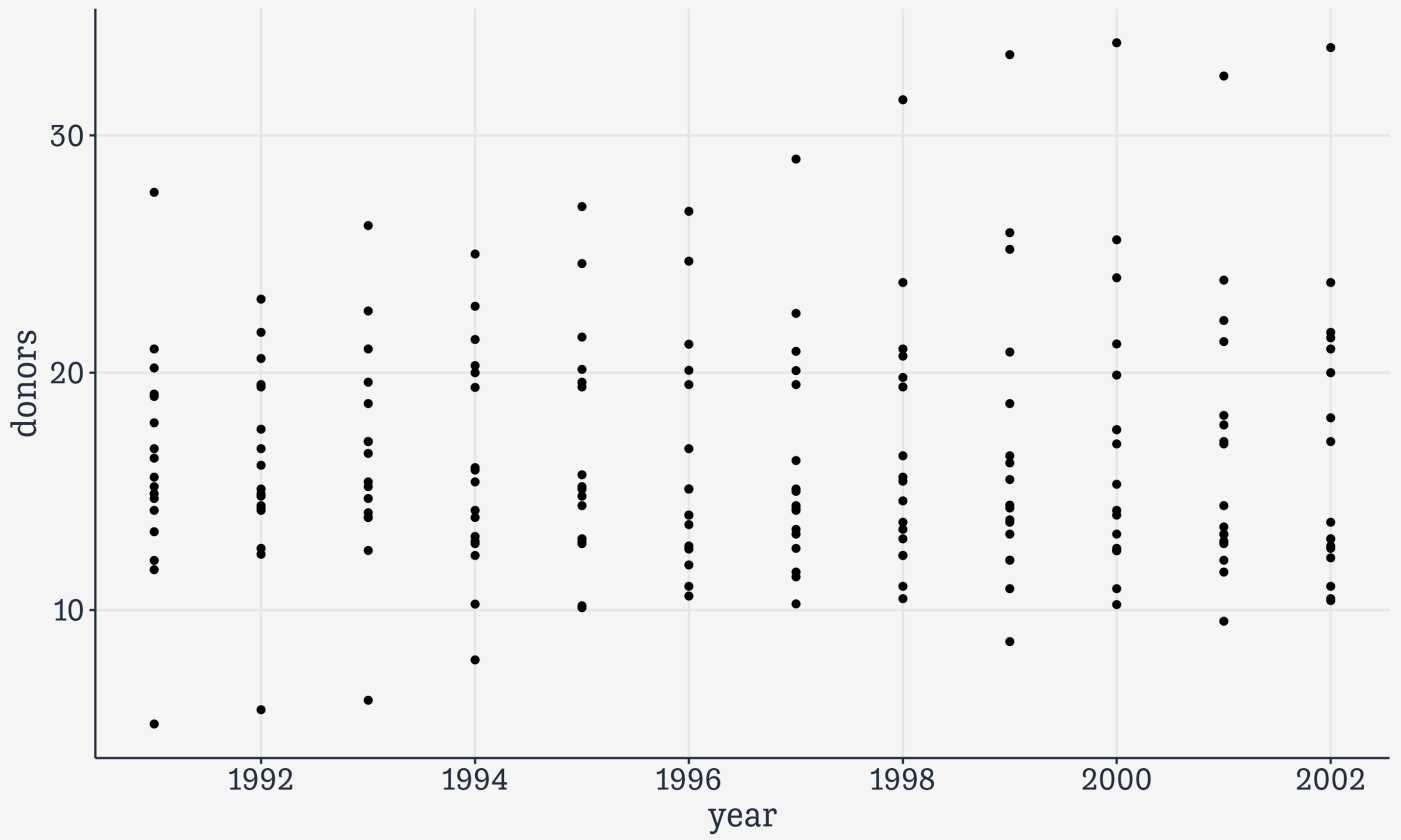
`organdata` is in the `socviz` package

```
organdata
```

```
# A tibble: 238 × 21
   country   year       donors   pop pop_dens   gdp gdp_lag health health_lag
   <chr>     <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
 1 Australia NA           NA    17065    0.220 16774   16591   1300       1224
 2 Australia 1991-01-01   12.1  17284    0.223 17171   16774   1379       1300
 3 Australia 1992-01-01   12.4  17495    0.226 17914   17171   1455       1379
 4 Australia 1993-01-01   12.5  17667    0.228 18883   17914   1540       1455
 5 Australia 1994-01-01   10.2  17855    0.231 19849   18883   1626       1540
 6 Australia 1995-01-01   10.2  18072    0.233 21079   19849   1737       1626
 7 Australia 1996-01-01   10.6  18311    0.237 21923   21079   1846       1737
 8 Australia 1997-01-01   10.3  18518    0.239 22961   21923   1948       1846
 9 Australia 1998-01-01   10.5  18711    0.242 24148   22961   2077       1948
10 Australia 1999-01-01    8.67 18926    0.244 25445   24148   2231       2077
# ℹ 228 more rows
# ℹ 12 more variables: pubhealth <dbl>, roads <dbl>, cerebvas <int>,
#   assault <int>, external <int>, txp_pop <dbl>, world <chr>, opt <chr>,
#   consent_law <chr>, consent_practice <chr>, consistent <chr>, ccode <chr>
```

First look

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

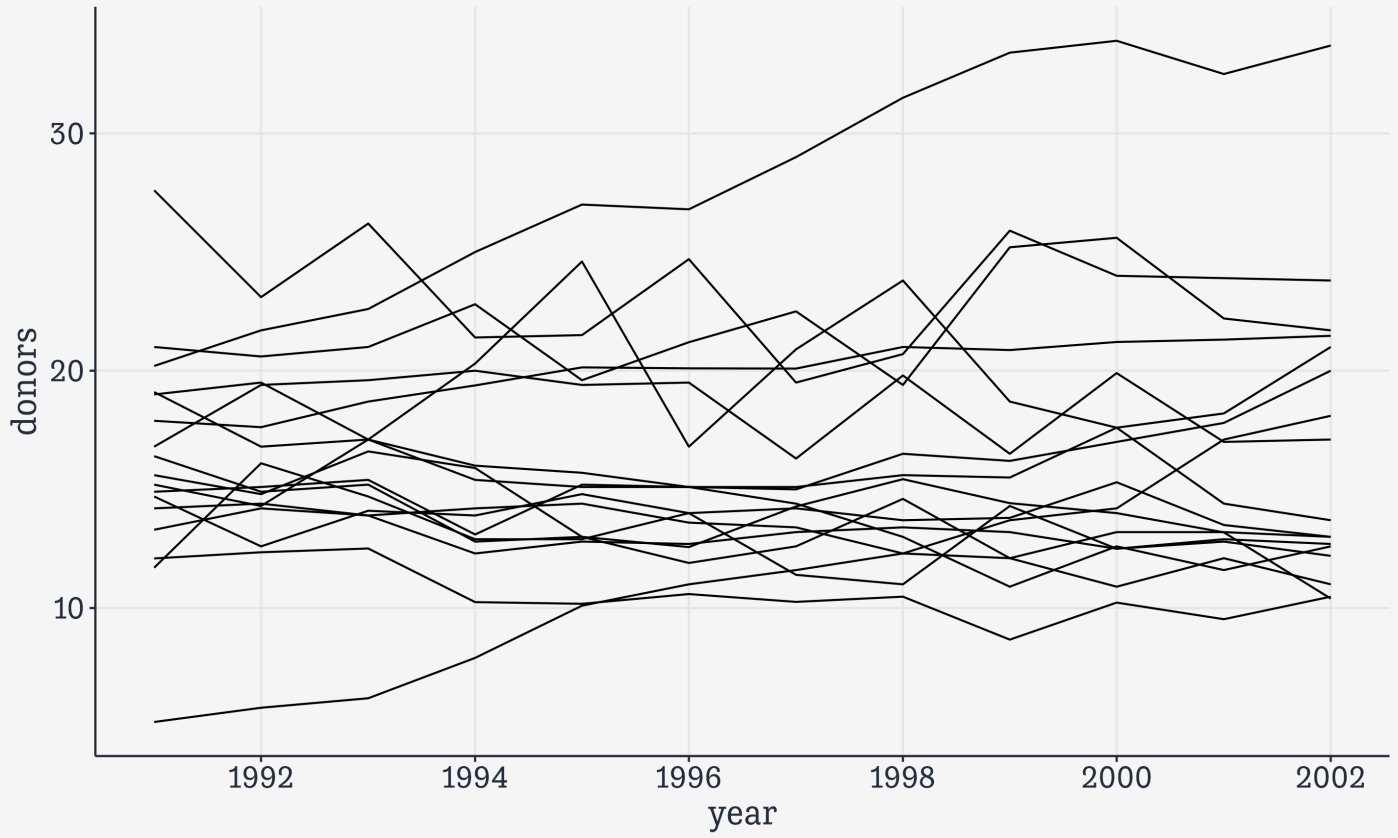


First look

```
p ← ggplot(data = organdata,
           mapping = aes(x = year, y = donors))
p + geom_line()
```

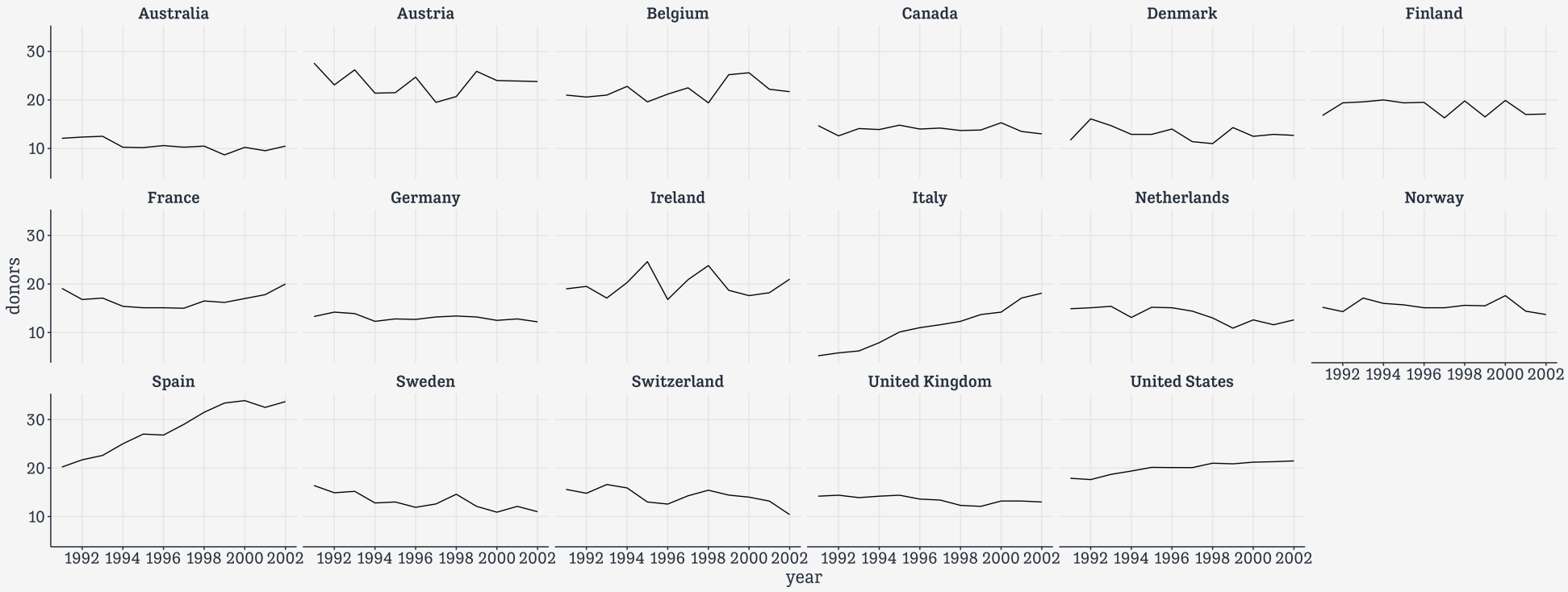
First look

```
p ← ggplot(data = organdata,
           mapping = aes(x = year, y = donors))
p + geom_line(aes(group = country))
```



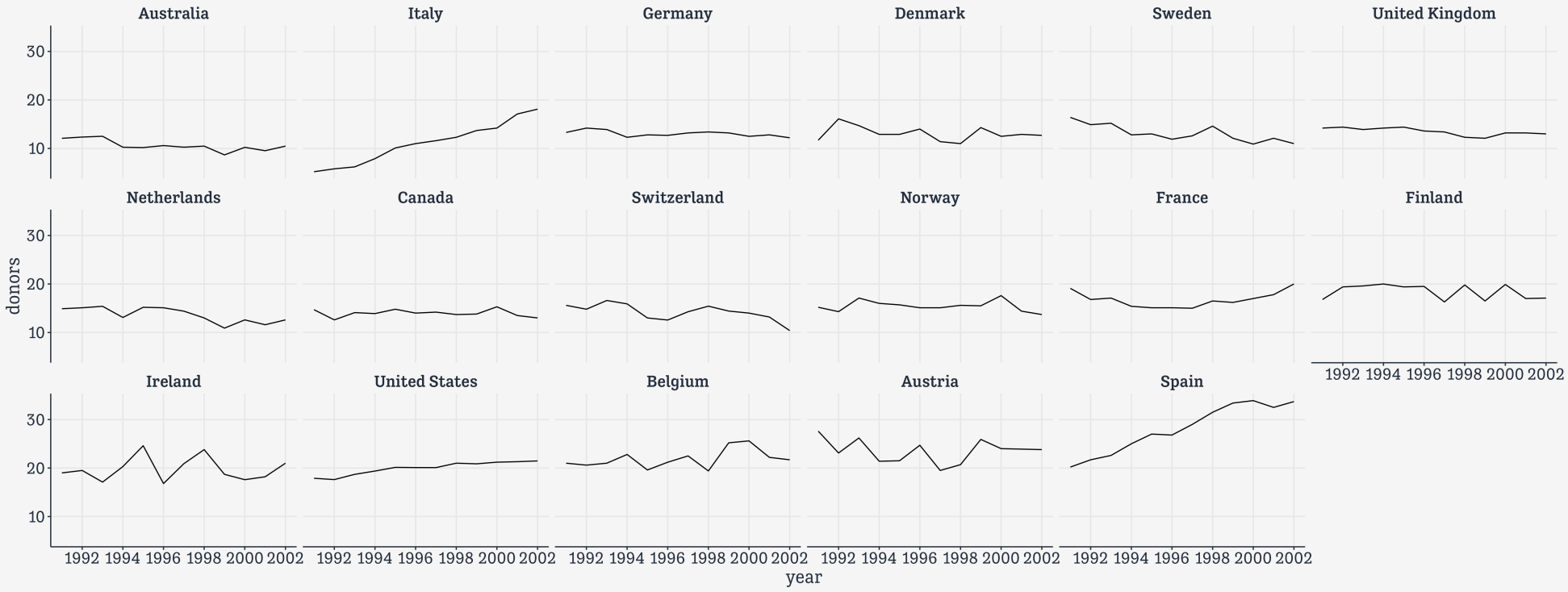
First look

```
p ← ggplot(data = organdata,
           mapping = aes(x = year, y = donors))
p + geom_line() +
  facet_wrap(~ country, nrow = 3)
```



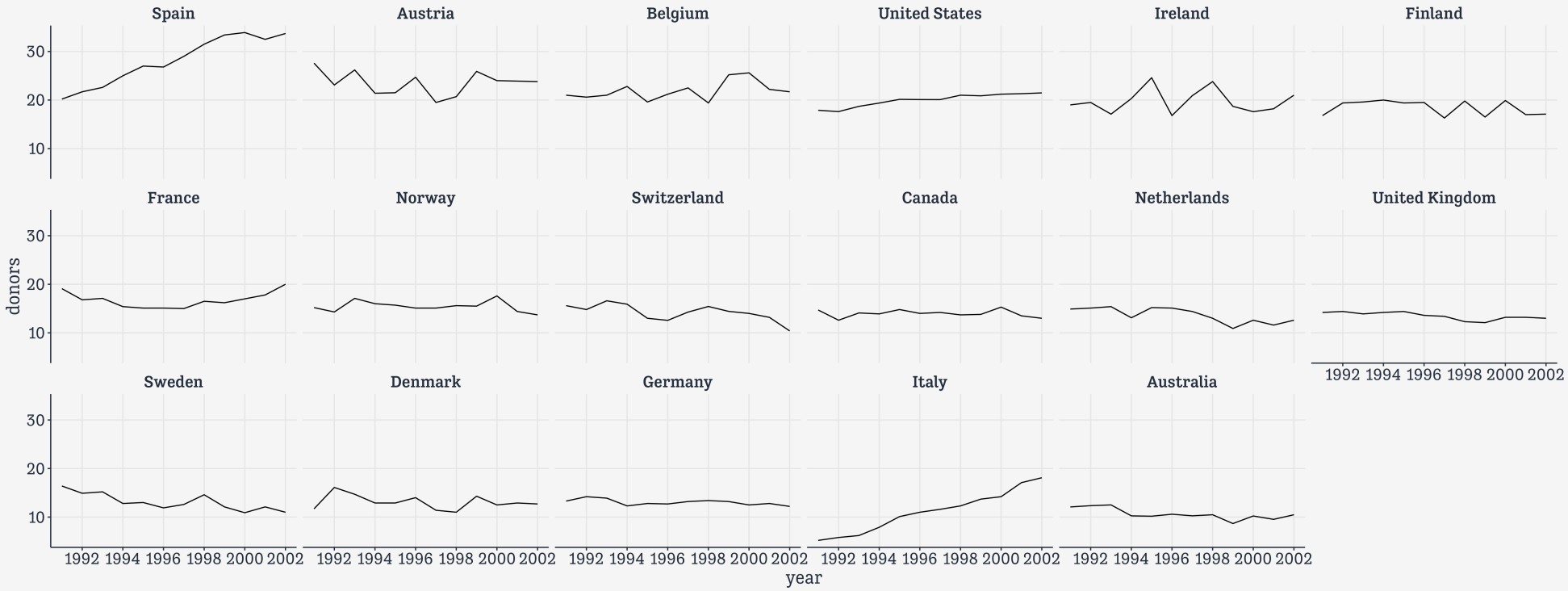
First look

```
p ← ggplot(data = organdata,
           mapping = aes(x = year, y = donors))
p + geom_line() +
  facet_wrap(~ reorder(country, donors, na.rm = TRUE), nrow = 3)
```



First look

```
p ← ggplot(data = organdata,
           mapping = aes(x = year, y = donors))
p + geom_line() +
  facet_wrap(~ reorder(country, -donors, na.rm = TRUE), nrow = 3)
```



Summarize better with `dplyr`

Summarize a bunch of variables

```
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 = mean(gdp, na.rm = TRUE),
              health_mean = mean(health, na.rm = TRUE),
              roads_mean = mean(roads, na.rm = TRUE),
              cerebvas_mean = mean(cerebvas, na.rm = TRUE))

head(by_country)
```

```
# A tibble: 6 × 8
# Groups:   consent_law [1]
  consent_law country     donors_mean donors_sd gdp_mean health_mean roads_mean
  <chr>       <chr>             <dbl>     <dbl>    <dbl>       <dbl>      <dbl>
1 Informed    Australia          10.6     1.14    22179.       1958.      105.
2 Informed    Canada             14.0     0.751   23711.       2272.      109.
3 Informed    Denmark            13.1     1.47    23722.       2054.      102.
4 Informed    Germany            13.0     0.611   22163.       2349.      113.
5 Informed    Ireland            19.8     2.48    20824.       1480.      118.
6 Informed    Netherlands        13.7     1.55    23013.       1993.       76.1
# ℹ 1 more variable: cerebvas_mean <dbl>
```

This works, but there's so much repetition! It's an open invitation to make mistakes copying and pasting

DRY:
Don’t Repeat Yourself

Use across() and where() instead

```
  by_country ← organdata ▷
    group_by(consent_law, country) ▷
      summarize(across(where(is.numeric),
                       list(mean = ~ mean(.x, na.rm = TRUE),
                            sd = ~ sd(.x, na.rm = TRUE))))
head(by_country)
```

```
# A tibble: 6 × 28
# Groups:   consent_law [1]
  consent_law country     donors_mean donors_sd pop_mean pop_sd pop_dens_mean
  <chr>       <chr>             <dbl>     <dbl>    <dbl>  <dbl>         <dbl>
1 Informed    Australia          10.6     1.14    18318.  831.          0.237
2 Informed    Canada             14.0     0.751   29608. 1193.          0.297
3 Informed    Denmark            13.1     1.47     5257.   80.6        12.2
4 Informed    Germany            13.0     0.611   80255. 5158.         22.5
5 Informed    Ireland            19.8     2.48     3674.  132.          5.23
6 Informed    Netherlands        13.7     1.55    15548.  373.         37.4
# ℹ 21 more variables: pop_dens_sd <dbl>, gdp_mean <dbl>, gdp_sd <dbl>,
#   gdp_lag_mean <dbl>, gdp_lag_sd <dbl>, health_mean <dbl>, health_sd <dbl>,
#   health_lag_mean <dbl>, health_lag_sd <dbl>, pubhealth_mean <dbl>,
#   pubhealth_sd <dbl>, roads_mean <dbl>, roads_sd <dbl>, cerebvas_mean <dbl>,
#   cerebvas_sd <dbl>, assault_mean <dbl>, assault_sd <dbl>,
#   external_mean <dbl>, external_sd <dbl>, txp_pop_mean <dbl>,
#   txp_pop_sd <dbl>
```

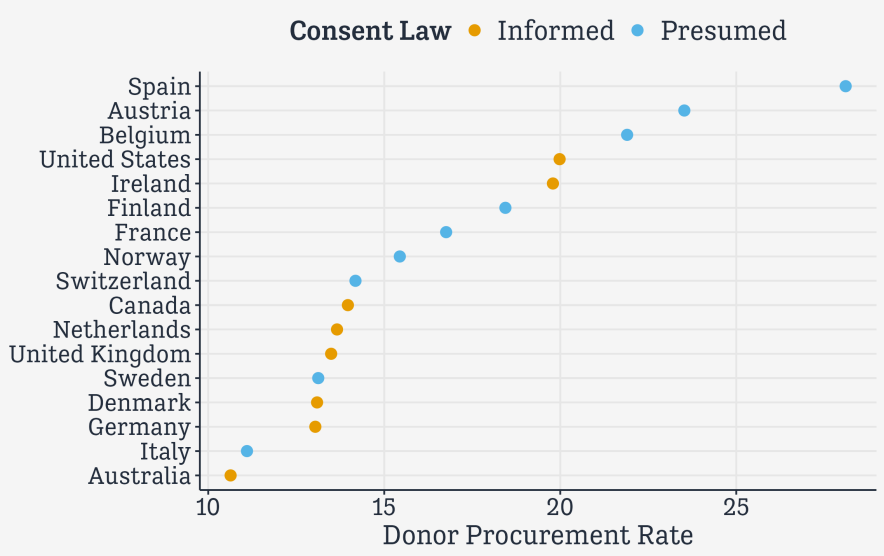
Use `across()` and `where()` instead

```
by_country ← organdata ▷
  group_by(consent_law, country) ▷
    summarize(across(where(is.numeric),
                     list(mean = ~ mean(.x, na.rm = TRUE),
                          sd = ~ sd(.x, na.rm = TRUE))),
              .groups = "drop")
head(by_country)
```

```
# A tibble: 6 × 28
  consent_law country     donors_mean donors_sd pop_mean pop_sd pop_dens_mean
  <chr>       <chr>             <dbl>     <dbl>    <dbl>  <dbl>         <dbl>
1 Informed    Australia          10.6     1.14    18318.  831.          0.237
2 Informed    Canada             14.0     0.751   29608. 1193.          0.297
3 Informed    Denmark            13.1     1.47     5257.   80.6        12.2
4 Informed    Germany            13.0     0.611   80255. 5158.         22.5
5 Informed    Ireland            19.8     2.48     3674.  132.          5.23
6 Informed    Netherlands        13.7     1.55    15548.  373.         37.4
# ℹ 21 more variables: pop_dens_sd <dbl>, gdp_mean <dbl>, gdp_sd <dbl>,
#   gdp_lag_mean <dbl>, gdp_lag_sd <dbl>, health_mean <dbl>, health_sd <dbl>,
#   health_lag_mean <dbl>, health_lag_sd <dbl>, pubhealth_mean <dbl>,
#   pubhealth_sd <dbl>, roads_mean <dbl>, roads_sd <dbl>, cerebvas_mean <dbl>,
#   cerebvas_sd <dbl>, assault_mean <dbl>, assault_sd <dbl>,
#   external_mean <dbl>, external_sd <dbl>, txp_pop_mean <dbl>,
#   txp_pop_sd <dbl>
```

Plot our summary data

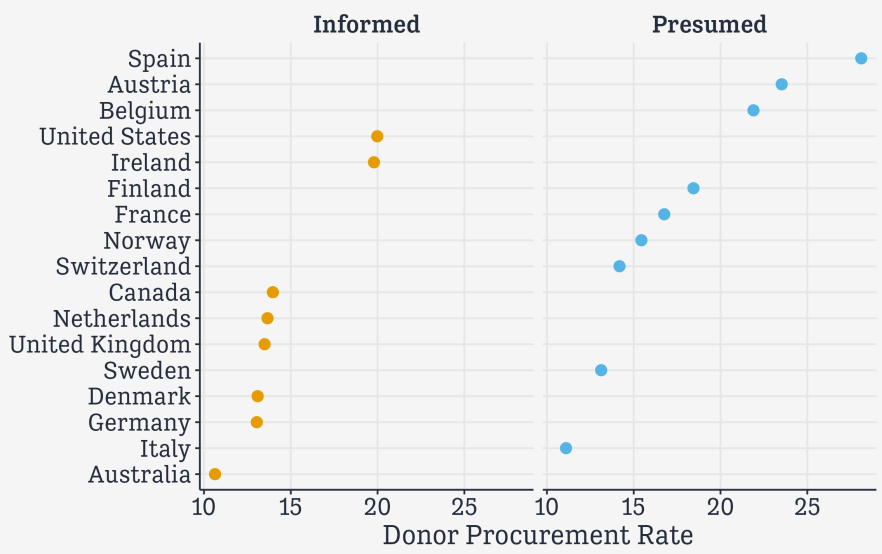
```
by_country ▷
  ggplot(mapping =
           aes(x = donors_mean,
               y = reorder(country, donors_mean
               color = consent_law)) +
  geom_point(size=3) +
  labs(x = "Donor Procurement Rate",
       y = NULL,
       color = "Consent Law")
```



What about faceting it instead?

The problem is that countries can only be in one Consent Law category.

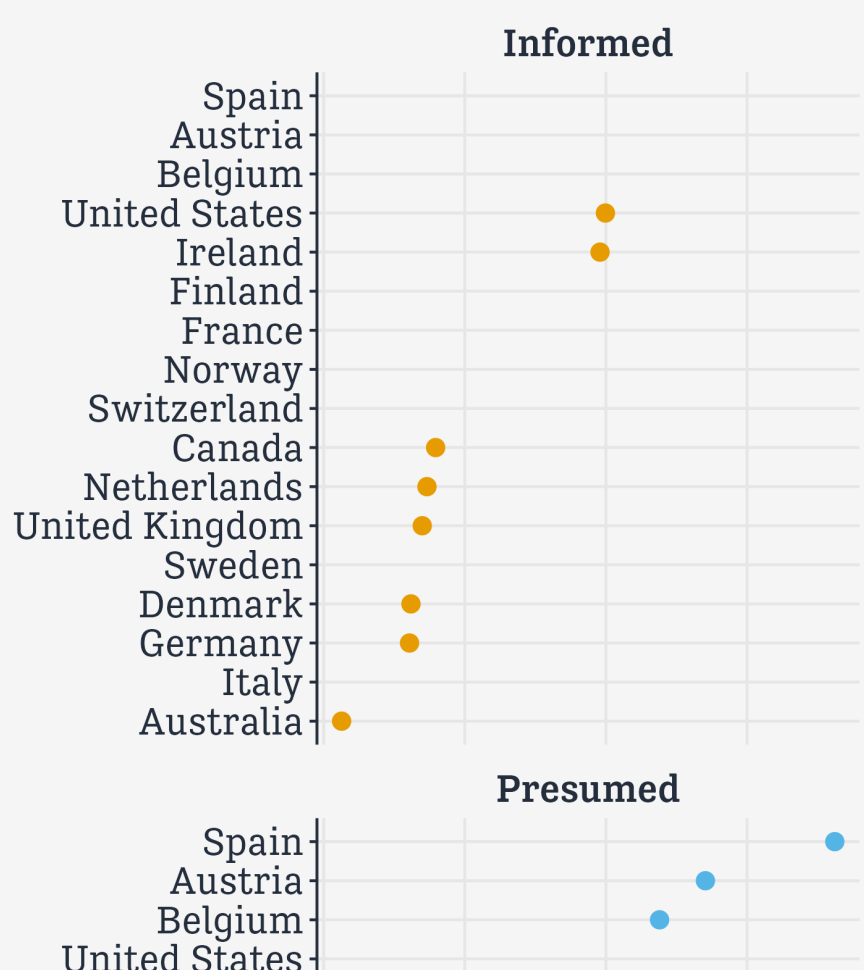
```
by_country ▷
  ggplot(mapping =
           aes(x = donors_mean,
               y = reorder(country, donors_mean
               color = consent_law)) +
  geom_point(size=3) +
  guides(color = "none") +
  facet_wrap(~ consent_law) +
  labs(x = "Donor Procurement Rate",
       y = NULL,
       color = "Consent Law")
```



What about faceting it instead?

Restricting to one column doesn't fix it.

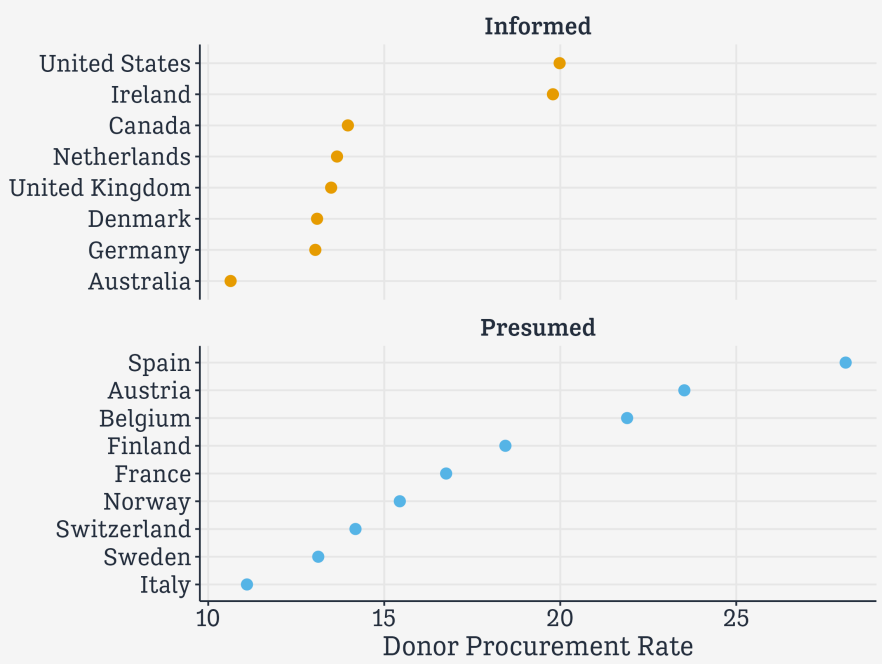
```
by_country ▷
  ggplot(mapping =
           aes(x = donors_mean,
               y = reorder(country, donors_mean
               color = consent_law)) +
  geom_point(size=3) +
  guides(color = "none") +
  facet_wrap(~ consent_law, ncol = 1) +
  labs(x = "Donor Procurement Rate",
       y = NULL,
       color = "Consent Law")
```



Allow the y-scale to vary

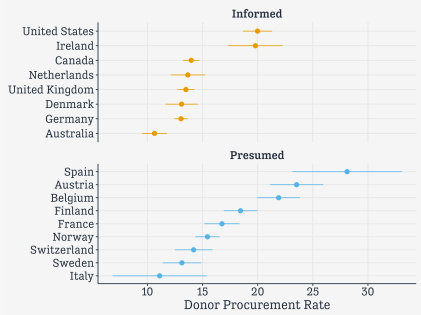
Normally the point of a facet is to preserve comparability between panels by not allowing the scales to vary. But for categorical measures it can be useful to allow this.

```
by_country ▷
  ggplot(mapping =
           aes(x = donors_mean,
               y = reorder(country, donors_mean
               color = consent_law)) +
  geom_point(size=3) +
  guides(color = "none") +
  facet_wrap(~ consent_law,
             ncol = 1,
             scales = "free_y") +
  labs(x = "Donor Procurement Rate",
       y = NULL,
       color = "Consent Law")
```



Again, these methods are general

```
by_country ▷
  ggplot(mapping =
           aes(x = dor
               y = rec
               color =
  geom_pointrange(mapp
                  ae

  guides(color = "none
  facet_wrap(~ consent
             ncol = 1,
             scales =
  labs(x = "Donor Proc
       y = NULL,
       color = "Conser
```



Plot text directly

geom_text() for