Gov 50: 5. Data Wrangling and Barplots

Matthew Blackwell

Harvard University

Roadmap

1. Operating on rows
2. Operating on columns
3. Operating on groups
4. Creating barplots

Local news data

- How does station ownership affect local news coverage?
- Martin and McCrain (2019) use data on local news at TV stations before and after a large acquisition by a conglomorate.

Variable	Description
`callsign`	Callsign of the station
`affiliation`	Network affiliation of the station
`date`	Airdate of news
`weekday`	Day of the week of airdate
`ideology`	Measure of news slant (bigger is more conservative)
`national_politics`	Avg proportion of segments on national politics
`local_politics`	Avg proportion of segments on national politics
`sinclair2017`	Station acquired by Sinclair group in Sept 2017
`post`	Date is before/after acquisition (0/1)

```r
library(gov50data)
data(news)
news
```

```
## # A tibble: 3,137 x 10
##    callsign affil~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon      NA      0.0286  0.0190       0
##  2 KTAB     CBS     2017-06-05 Mon      NA      0.0286  0.0190       0
##  3 KXVA     FOX     2017-06-05 Mon      NA      0.0393  0.0262       0
##  4 KPAX     CBS     2017-06-06 Tue      NA      0.00357 0.194        0
##  5 KTAB     CBS     2017-06-06 Tue      NA      0.0945  0.109        0
##  6 KECI     NBC     2017-06-07 Wed       0.0655 0.225   0.148        1
##  7 KPAX     CBS     2017-06-07 Wed       0.0853 0.283   0.123        0
##  8 KRBC     NBC     2017-06-07 Wed       0.0183 0.130   0.189        0
##  9 KTAB     CBS     2017-06-07 Wed       0.0850 0.0901  0.138        0
## 10 KTMF     ABC     2017-06-07 Wed       0.0842 0.152   0.129        0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: affiliation,
## #   2: ideology, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

1/ Operating on rows

slice()

slice() can give you a specific set of rows:

```
## first and third row
news |>
  slice(1, 3)
```

```
## # A tibble: 2 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-06-05 Mon          NA  0.0286  0.0190       0
## 2 KXVA     FOX      2017-06-05 Mon          NA  0.0393  0.0262       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

You can ask for a range of rows with `start:stop` syntax:

```
## first three rows
news |>
  slice(1:3)
```

```
## # A tibble: 3 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-06-05 Mon          NA  0.0286  0.0190       0
## 2 KTAB     CBS      2017-06-05 Mon          NA  0.0286  0.0190       0
## 3 KXVA     FOX      2017-06-05 Mon          NA  0.0393  0.0262       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

slice_max()

`slice_max(var, n = 5)` will return the top 5 observations on column `var`

```
news |>
  slice_max(ideology, n = 5)
```

```
## # A tibble: 5 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KAEF     ABC      2017-06-19 Mon       0.778  0.0823   0.179       1
## 2 WYDO     FOX      2017-07-19 Wed       0.580  0.126    0.121       1
## 3 KRCR     ABC      2017-10-03 Tue       0.566  0.123    0.192       1
## 4 KAEF     ABC      2017-10-18 Wed       0.496  0.0892   0.217       1
## 5 KBVU     FOX      2017-11-16 Thu       0.491  0.159    0.184       1
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

slice_min()

`slice_min(var, n = 5)` will return the bottom 5 observations on column `var`

```
news |>
  slice_min(ideology, n = 5)
```

```
## # A tibble: 5 x 10
##   callsign affili~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
## 1 KRBC     NBC      2017-10-19 Thu      -0.674  0.0731   0.161       0
## 2 WJHL     CBS      2017-12-08 Fri      -0.673  0.0364   0.206       0
## 3 KRBC     NBC      2017-10-18 Wed      -0.586  0.0470   0.135       0
## 4 KCVU     FOX      2017-06-22 Thu      -0.414  0.158    0.172       1
## 5 KRBC     NBC      2017-12-11 Mon      -0.365  0.0674   0.163       0
## # ... with 2 more variables: post <dbl>, month <ord>, and abbreviated
## #   variable names 1: affiliation, 2: ideology, 3: national_politics,
## #   4: local_politics, 5: sinclair2017
```

2/ Operating on columns

rename()

`rename(new_name = old_name)` renames the `old_name` variable to `new_name`

```
news |>
  rename(call_sign = callsign)
```

```
## # A tibble: 3,137 x 10
##    call_s~1 affil~2 date       weekday ideol~3 natio~4 local~5 sincl~6
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon      NA      0.0286   0.0190       0
##  2 KTAB     CBS     2017-06-05 Mon      NA      0.0286   0.0190       0
##  3 KXVA     FOX     2017-06-05 Mon      NA      0.0393   0.0262       0
##  4 KPAX     CBS     2017-06-06 Tue      NA      0.00357  0.194        0
##  5 KTAB     CBS     2017-06-06 Tue      NA      0.0945   0.109        0
##  6 KECI     NBC     2017-06-07 Wed       0.0655 0.225    0.148        1
##  7 KPAX     CBS     2017-06-07 Wed       0.0853 0.283    0.123        0
##  8 KRBC     NBC     2017-06-07 Wed       0.0183 0.130    0.189        0
##  9 KTAB     CBS     2017-06-07 Wed       0.0850 0.0901   0.138        0
## 10 KTMF     ABC     2017-06-07 Wed       0.0842 0.152    0.129        0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: call_sign,
## #   2: affiliation, 3: ideology, 4: national_politics,
## #   5: local_politics, 6: sinclair2017
```

mutate()

`mutate(new_var = fun(old_vars))` adds new columns that are functions of existing columns.

```r
news |>
  mutate(
    national_local_diff = national_politics - local_politics,
    national_politics_perc = national_politics * 100
  ) |>
  select(callsign, date, national_politics, local_politics,
         national_local_diff, national_politics_perc)
```

```
## # A tibble: 3,137 x 6
##    callsign date       national_politics local_politics national_local_diff national_politics_perc
##    <chr>    <date>                 <dbl>          <dbl>               <dbl>                  <dbl>
##  1 KRBC     2017-06-05           0.0286         0.0190             0.00952                   2.86
##  2 KTAB     2017-06-05           0.0286         0.0190             0.00952                   2.86
##  3 KXVA     2017-06-05           0.0393         0.0262             0.0131                    3.93
##  4 KPAX     2017-06-06           0.00357        0.194              -0.191                    0.357
##  5 KTAB     2017-06-06           0.0945         0.109              -0.0145                   9.45
##  6 KECI     2017-06-07           0.225          0.148              0.0761                   22.5
##  7 KPAX     2017-06-07           0.283          0.123              0.160                    28.3
##  8 KRBC     2017-06-07           0.130          0.189              -0.0589                  13.0
##  9 KTAB     2017-06-07           0.0901         0.138              -0.0476                   9.01
## 10 KTMF     2017-06-07           0.152          0.129              0.0229                   15.2
## # ... with 3,127 more rows
```

`if_else()`

`if_else(test_condition, yes, no)` allows us to create a vector that depends on a logical

New vector gets `yes` expression when `test_condition` is TRUE, `no` otherwise

```
news |>
  mutate(Ownership = if_else(sinclair2017 == 1,
                             "Acquired by Sinclair",
                             "Not Acquired")) |>
  select(callsign, affiliation, date, Ownership)
```

```
## # A tibble: 3,137 x 4
##    callsign affiliation date       Ownership
##    <chr>    <chr>       <date>     <chr>
##  1 KRBC     NBC         2017-06-05 Not Acquired
##  2 KTAB     CBS         2017-06-05 Not Acquired
##  3 KXVA     FOX         2017-06-05 Not Acquired
##  4 KPAX     CBS         2017-06-06 Not Acquired
##  5 KTAB     CBS         2017-06-06 Not Acquired
##  6 KECI     NBC         2017-06-07 Acquired by Sinclair
##  7 KPAX     CBS         2017-06-07 Not Acquired
##  8 KRBC     NBC         2017-06-07 Not Acquired
##  9 KTAB     CBS         2017-06-07 Not Acquired
## 10 KTMF     ABC         2017-06-07 Not Acquired
## # ... with 3,127 more rows
```

3/ Operating on groups

group_by()

`group_by(var)` divides the data into groups based on the `var` variable.

Doesn't change data yet, but subsequent operations will by `var`.

```
news |>
  group_by(month)
```

```
## # A tibble: 3,137 x 10
## # Groups:   month [7]
##    callsign affil~1 date       weekday ideol~2 natio~3 local~4 sincl~5
##    <chr>    <chr>   <date>     <ord>     <dbl>   <dbl>   <dbl>   <dbl>
##  1 KRBC     NBC     2017-06-05 Mon      NA      0.0286   0.0190       0
##  2 KTAB     CBS     2017-06-05 Mon      NA      0.0286   0.0190       0
##  3 KXVA     FOX     2017-06-05 Mon      NA      0.0393   0.0262       0
##  4 KPAX     CBS     2017-06-06 Tue      NA      0.00357  0.194        0
##  5 KTAB     CBS     2017-06-06 Tue      NA      0.0945   0.109        0
##  6 KECI     NBC     2017-06-07 Wed       0.0655 0.225    0.148        1
##  7 KPAX     CBS     2017-06-07 Wed       0.0853 0.283    0.123        0
##  8 KRBC     NBC     2017-06-07 Wed       0.0183 0.130    0.189        0
##  9 KTAB     CBS     2017-06-07 Wed       0.0850 0.0901   0.138        0
## 10 KTMF     ABC     2017-06-07 Wed       0.0842 0.152    0.129        0
## # ... with 3,127 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: affiliation,
## #   2: ideology, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

summarize()

`summarize(sum_var = fun(curr_var))` calculates summaries of variables by groups.

Ideological slant by weekday

```
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  )
```

```
## # A tibble: 7 x 2
##   month slant_mean
##   <ord>      <dbl>
## 1 Jun       0.0786
## 2 Jul       0.103
## 3 Aug       0.105
## 4 Sep       0.0751
## 5 Oct       0.0862
## 6 Nov       0.0972
## 7 Dec       0.0774
```

Summaries by ownership and pre/post

```
news |>
  group_by(sinclair2017, post) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE),
    national_mean = mean(national_politics, na.rm = TRUE)
  )
```

```
## # A tibble: 4 x 4
## # Groups:   sinclair2017 [2]
##   sinclair2017  post slant_mean national_mean
##          <dbl> <dbl>      <dbl>         <dbl>
## 1            0     0     0.100          0.118
## 2            0     1     0.0768         0.107
## 3            1     0     0.0936         0.124
## 4            1     1     0.0938         0.144
```

Summarize across types of variables

`across()` will apply a summary function across many variables

```
news |>
  group_by(sinclair2017, post) |>
  summarize(
    across(where(is.numeric), mean, na.rm = TRUE),
  )
```

```
## # A tibble: 4 x 5
## # Groups:   sinclair2017 [2]
##   sinclair2017  post ideology national_politics local_politics
##          <dbl> <dbl>    <dbl>             <dbl>          <dbl>
## 1            0     0   0.100              0.118          0.158
## 2            0     1   0.0768             0.107          0.150
## 3            1     0   0.0936             0.124          0.170
## 4            1     1   0.0938             0.144          0.147
```

`kable()` to produce nice tables

```
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  ) |>
  knitr::kable()
```

month	slant_mean
Jun	0.079
Jul	0.103
Aug	0.105
Sep	0.075
Oct	0.086
Nov	0.097
Dec	0.077

Giving nicer column names

```
news |>
  group_by(month) |>
  summarize(
    slant_mean = mean(ideology, na.rm = TRUE)
  ) |>
  knitr::kable(col.names = c("Month", "Avg. Slant"))
```

Month	Avg. Slant
Jun	0.079
Jul	0.103
Aug	0.105
Sep	0.075
Oct	0.086
Nov	0.097
Dec	0.077

Producing a table of counts of a categorical variable

```
news |>
  group_by(affiliation) |>
  summarize(n = n())
```

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           863
## 2 CBS           807
## 3 FOX           662
## 4 NBC           805
```

Helper function `count()`

`count()` does the same thing:

```
news |>
  count(affiliation)
```

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           863
## 2 CBS           807
## 3 FOX           662
## 4 NBC           805
```

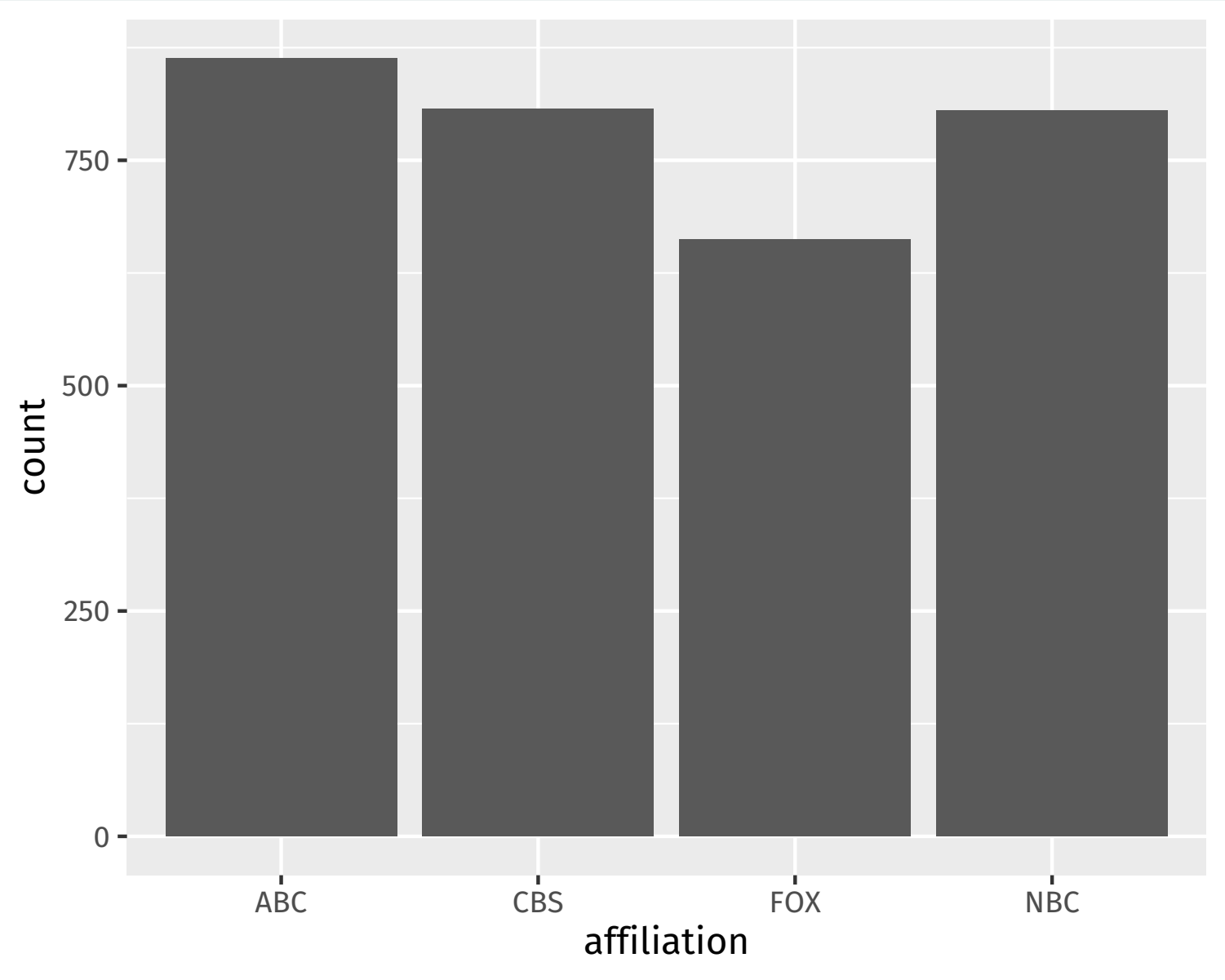
4/ Creating barplots

Combining our skills

Let's combine our tools to produce a bar plot with `geom_bar()`

By default, bar plots take a single variable and show the number of observations in each category.

```
ggplot(news, mapping = aes(x = affiliation)) +
  geom_bar()
```



Barplots of non-counts

Barplots can represent a lot beyond counts, including variables in our dataset or group summaries.

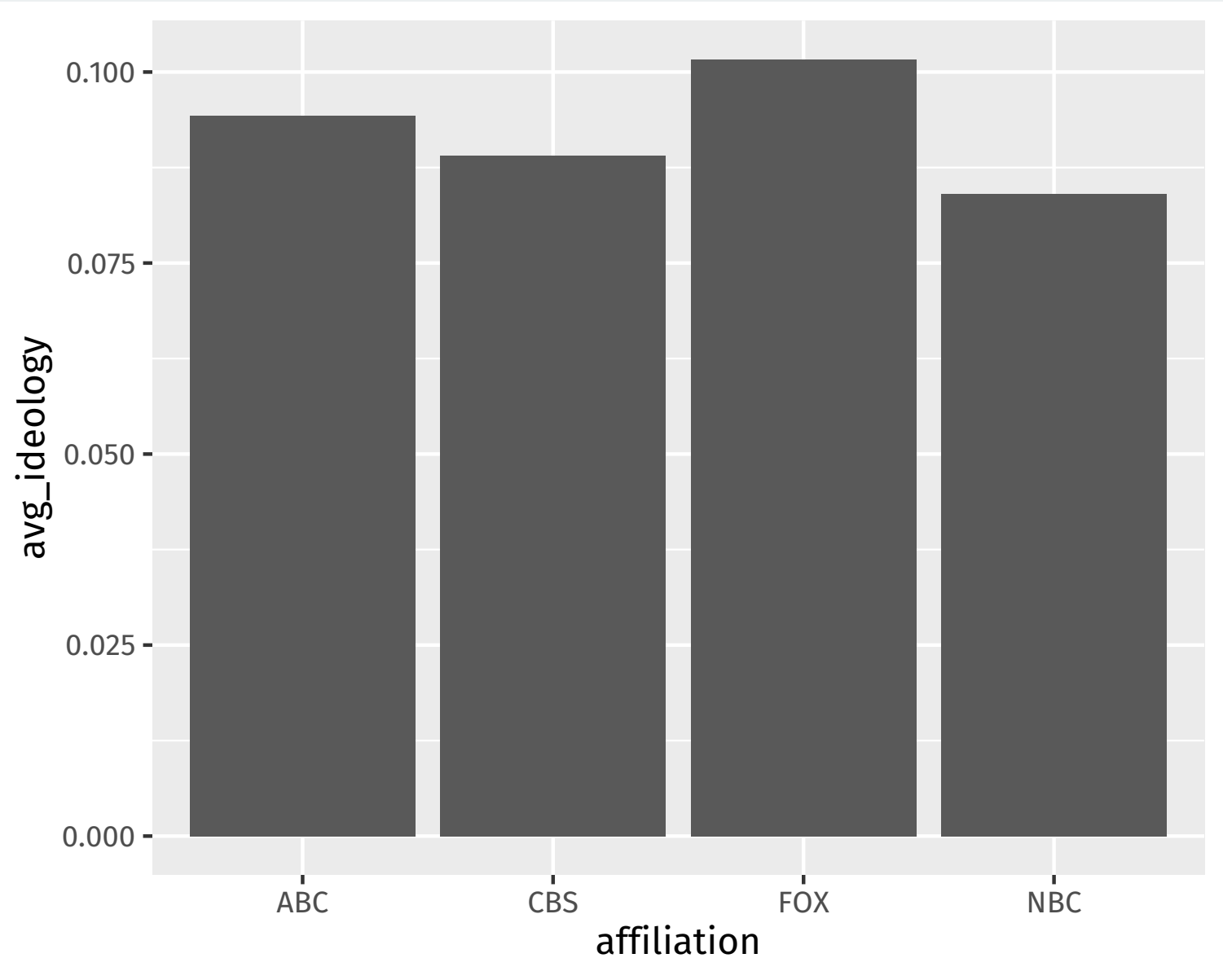
When the height of the bar is another variable in our data and not just a count, we set the x and y aesthetics and use `geom_col()` instead of `geom_bar()`.

Let's create a group summary:

```
aff_ideology_means <- news |>
  group_by(affiliation) |>
  summarize(avg_ideology = mean(ideology, na.rm = TRUE))
aff_ideology_means
```

```
## # A tibble: 4 x 2
##   affiliation avg_ideology
##   <chr>              <dbl>
## 1 ABC               0.0943
## 2 CBS               0.0891
## 3 FOX               0.102
## 4 NBC               0.0841
```

```
ggplot(aff_ideology_means, aes(x = affiliation, y = avg_ideology)) +
  geom_col()
```

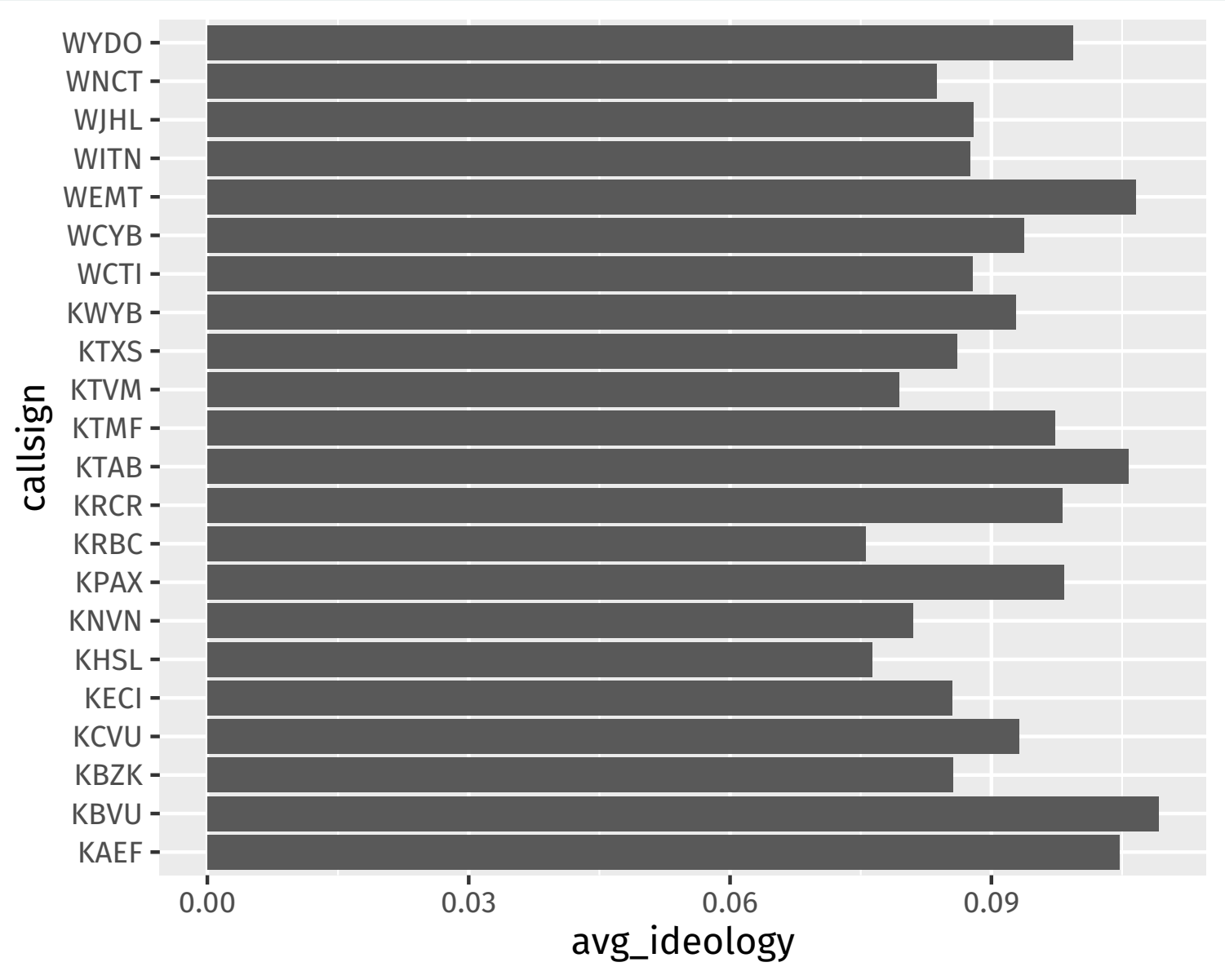

0.100
0.075
0.050
0.025
0.000
avg_ideology
ABC
CBS
FOX
NBC
affiliation

A more complicated example

Let's create a barplot that shows the top 10 stations in terms of slant. First, let's get the data:

```
station_ideology <- news |>
  group_by(callsign, affiliation) |>
  summarize(avg_ideology = mean(ideology, na.rm = TRUE)) |>
  slice_max(avg_ideology, n = 20)
```

```
ggplot(station_ideology, aes(x = avg_ideology, y = callsign)) +
  geom_col()
```

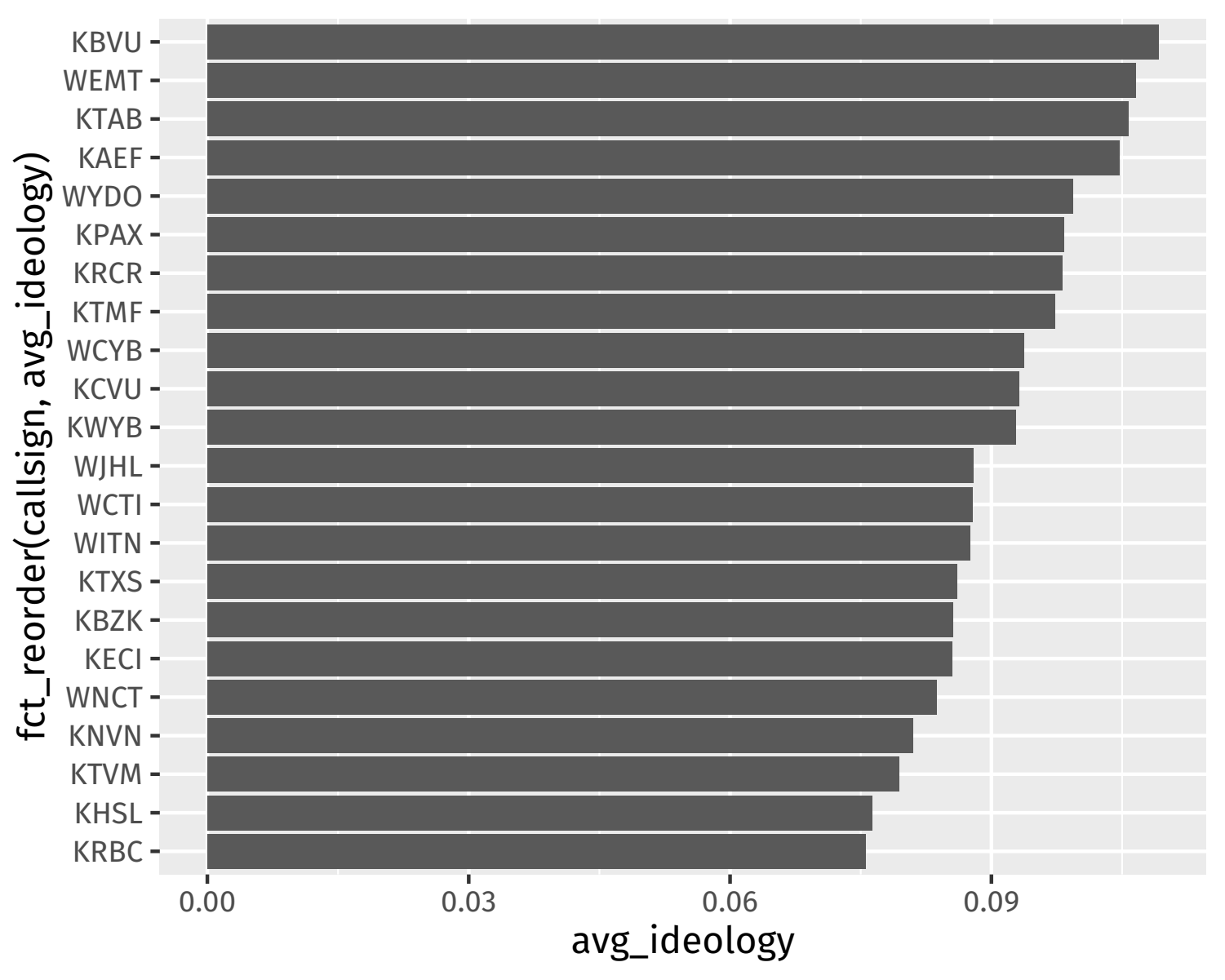


How do we reorder the stations?

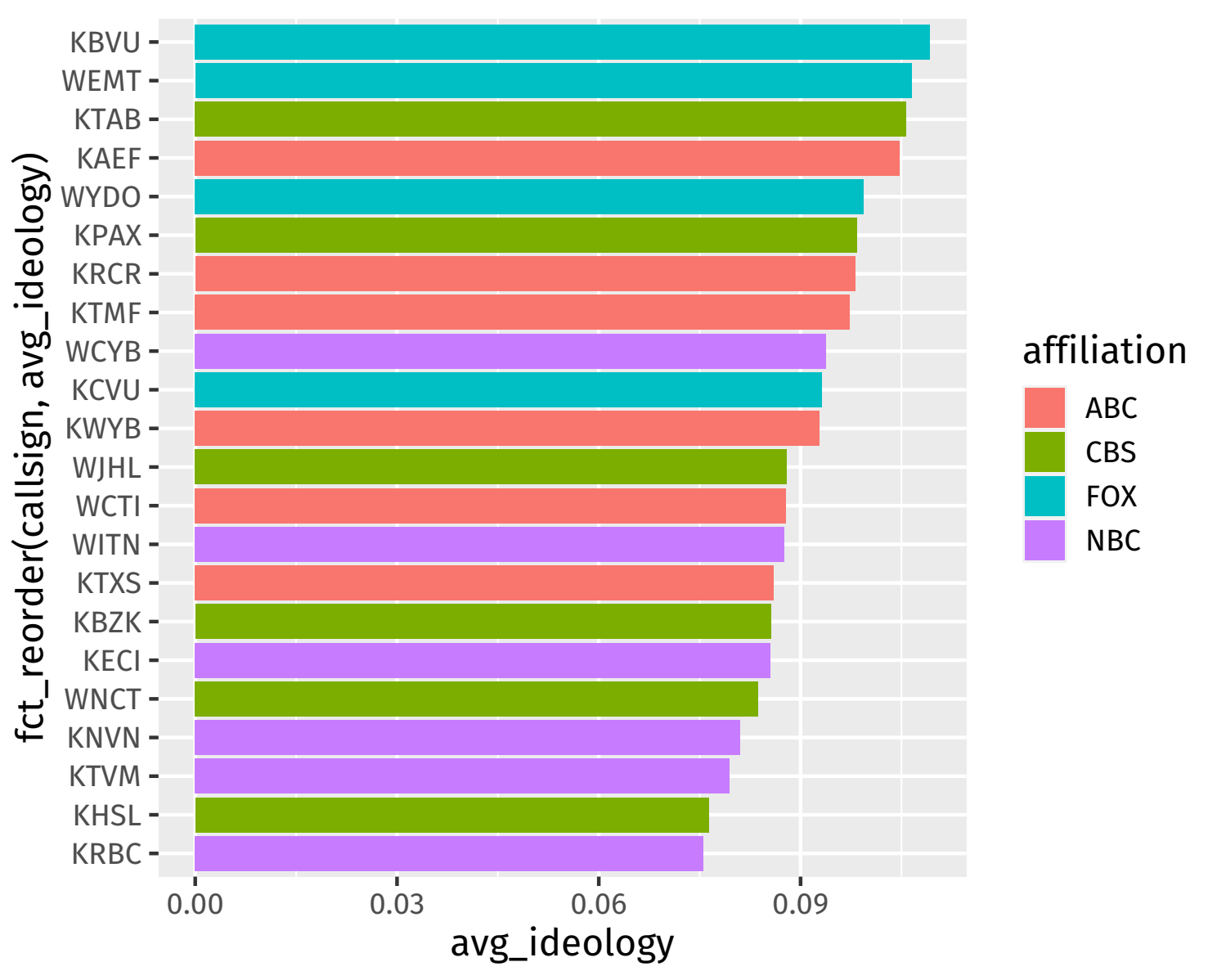
We would like to order the stations by ideology.

`fct_reorder(group, order_var)` function (loaded with tidyverse) will reorder the groups by the order bar (low to high). Easiest to put this in the mapping.

```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col()
```



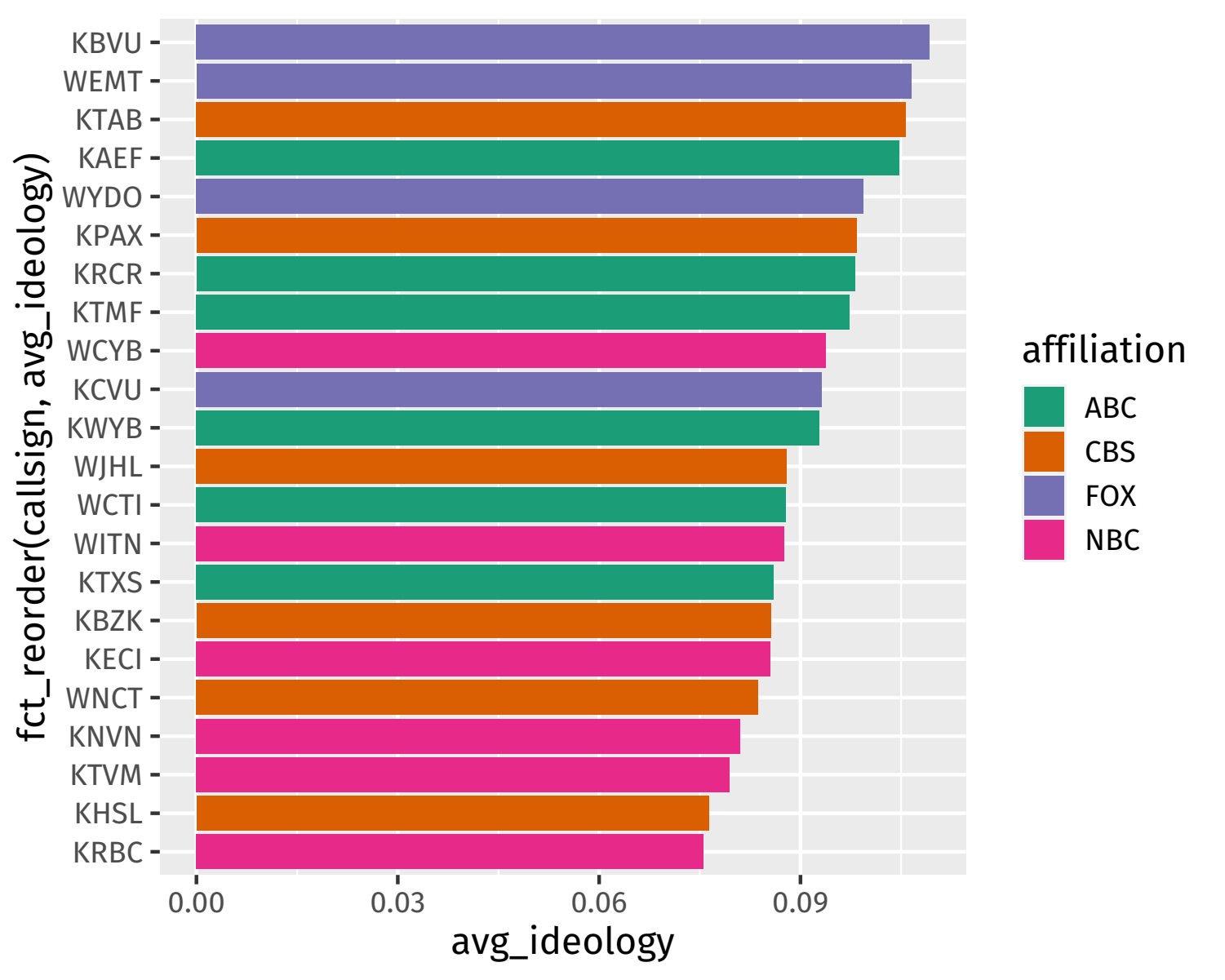
```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation))
```



Setting the color palette

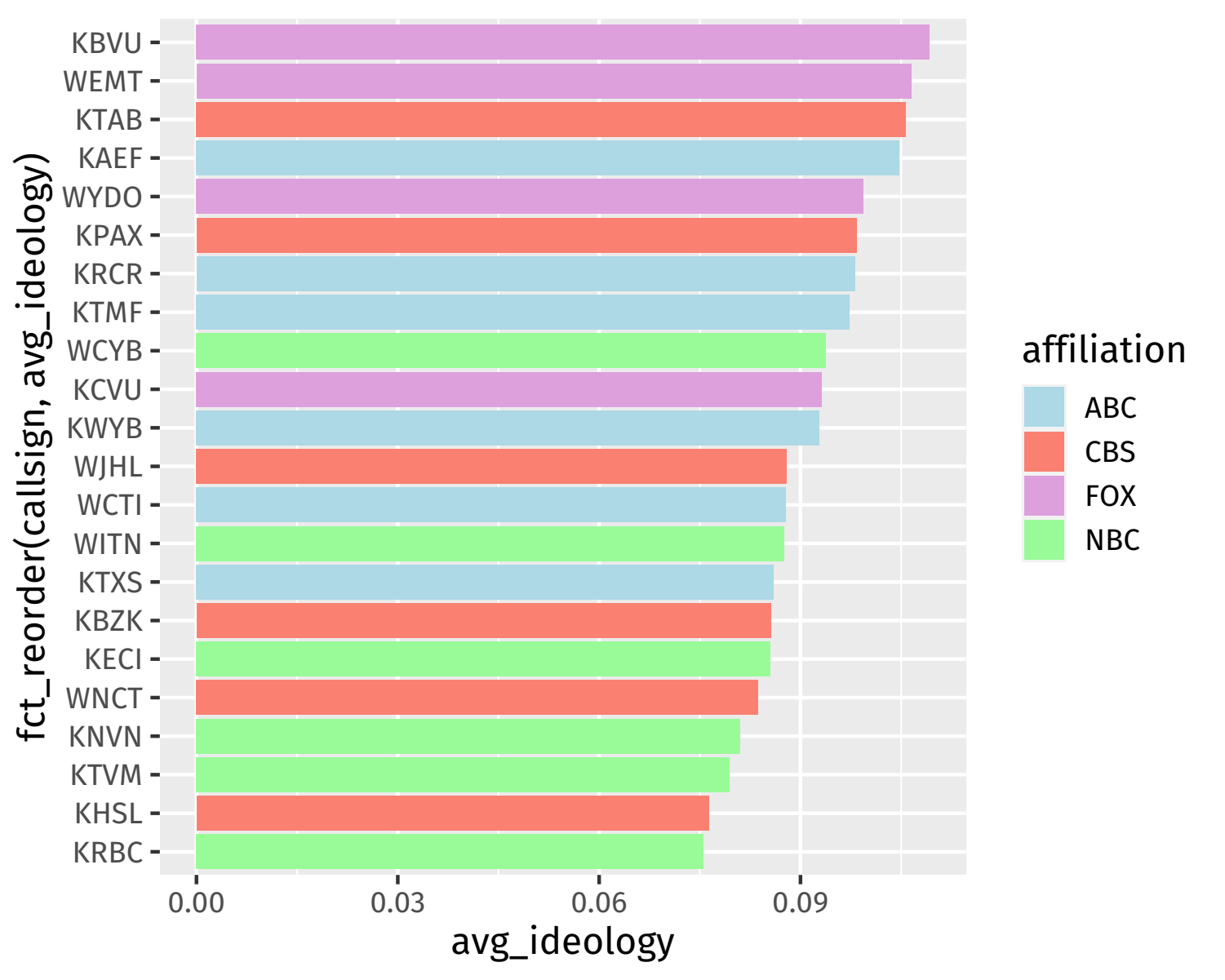
We can use color palettes from a project called ColorBrewer

```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_brewer(palette = "Dark2")
```


KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
fct_reorder(callsign, avg_ideology)
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC

Manually setting the color palette

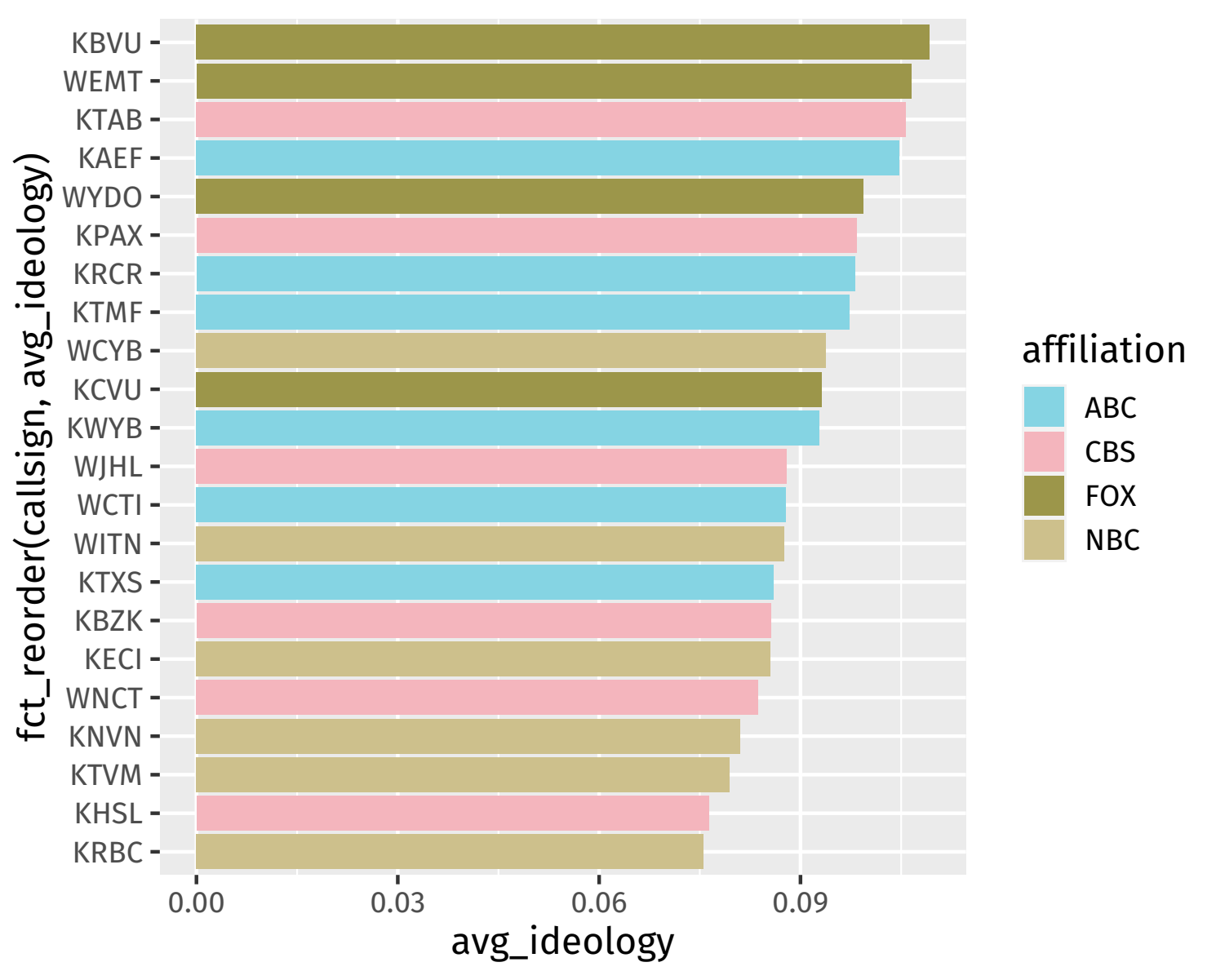
```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = c(ABC = "lightblue",
                               CBS = "salmon",
                               FOX = "plum",
                               NBC = "palegreen"))
```


KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
fct_reorder(callsign, avg_ideology)
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC

Fun with colors

Other packages provide more palettes:

```
library(wesanderson)
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = wes_palette("Moonrise3"))
```


KBVU
WEMT
KTAB
KAEF
WYDO
KPAX
KRCR
KTMF
WCYB
KCVU
KWYB
WJHL
WCTI
WITN
KTXS
KBZK
KECI
WNCT
KNVN
KTVM
KHSL
KRBC
fct_reorder(callsign, avg_ideology)
0.00
0.03
0.06
0.09
avg_ideology
affiliation
ABC
CBS
FOX
NBC