Gov 50: 5. Data Wrangling and Barplots

Matthew Blackwell

Harvard University

Roadmap

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)
news <- na.omit(news) ## drop missing data
news
```

```
## # A tibble: 2,560 x 10
##    callsign affiliation date       weekday  ideology
##    <chr>    <chr>       <date>     <ord>       <dbl>
##  1 KECI     NBC         2017-06-07 Wed      0.0655
##  2 KPAX     CBS         2017-06-07 Wed      0.0853
##  3 KRBC     NBC         2017-06-07 Wed      0.0183
##  4 KTAB     CBS         2017-06-07 Wed      0.0850
##  5 KTMF     ABC         2017-06-07 Wed      0.0842
##  6 KTXS     ABC         2017-06-07 Wed     -0.000488
##  7 KAEF     ABC         2017-06-08 Thu      0.0426
##  8 KBVU     FOX         2017-06-08 Thu     -0.0860
##  9 KECI     NBC         2017-06-08 Thu      0.0902
## 10 KPAX     CBS         2017-06-08 Thu      0.0668
## # i 2,550 more rows
## # i 5 more variables: national_politics <dbl>,
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>
```

1/ Operating on rows

slice()

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

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

```
## # A tibble: 2 x 10
##   callsign affiliation date       weekday ideology
##   <chr>    <chr>       <date>     <ord>      <dbl>
## 1 KECI     NBC         2017-06-07 Wed       0.0655
## 2 KRBC     NBC         2017-06-07 Wed       0.0183
## # i 5 more variables: national_politics <dbl>,
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>
```

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 affiliation date       weekday ideology
##   <chr>    <chr>       <date>     <ord>      <dbl>
## 1 KECI     NBC         2017-06-07 Wed       0.0655
## 2 KPAX     CBS         2017-06-07 Wed       0.0853
## 3 KRBC     NBC         2017-06-07 Wed       0.0183
## # i 5 more variables: national_politics <dbl>,
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>
```

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 affiliation date       weekday ideology
##   <chr>    <chr>       <date>     <ord>      <dbl>
## 1 KAEF     ABC         2017-06-19 Mon        0.778
## 2 WYDO     FOX         2017-07-19 Wed        0.580
## 3 KRCR     ABC         2017-10-03 Tue        0.566
## 4 KAEF     ABC         2017-10-18 Wed        0.496
## 5 KBVU     FOX         2017-11-16 Thu        0.491
## # i 5 more variables: national_politics <dbl>,
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>
```

slice_min()

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

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

```
## # A tibble: 5 x 10
##   callsign affiliation date       weekday ideology
##   <chr>    <chr>       <date>     <ord>      <dbl>
## 1 KRBC     NBC         2017-10-19 Thu       -0.674
## 2 WJHL     CBS         2017-12-08 Fri       -0.673
## 3 KRBC     NBC         2017-10-18 Wed       -0.586
## 4 KCVU     FOX         2017-06-22 Thu       -0.414
## 5 KRBC     NBC         2017-12-11 Mon       -0.365
## # i 5 more variables: national_politics <dbl>,
## #   local_politics <dbl>, sinclair2017 <dbl>, post <dbl>,
## #   month <ord>
```

2/ 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`.

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

```
## # A tibble: 2,560 x 10
## # Groups:   month [7]
##    callsign affiliation date       weekday  ideology national_politics
##    <chr>    <chr>       <date>     <ord>       <dbl>             <dbl>
##  1 KECI     NBC         2017-06-07 Wed      0.0655              0.225 
##  2 KPAX     CBS         2017-06-07 Wed      0.0853              0.283 
##  3 KRBC     NBC         2017-06-07 Wed      0.0183              0.130 
##  4 KTAB     CBS         2017-06-07 Wed      0.0850              0.0901
##  5 KTMF     ABC         2017-06-07 Wed      0.0842              0.152 
##  6 KTXS     ABC         2017-06-07 Wed     -0.000488            0.0925
##  7 KAEF     ABC         2017-06-08 Thu      0.0426              0.213 
##  8 KBVU     FOX         2017-06-08 Thu     -0.0860              0.169 
##  9 KECI     NBC         2017-06-08 Thu      0.0902              0.276 
## 10 KPAX     CBS         2017-06-08 Thu      0.0668              0.305 
## # i 2,550 more rows
## # i 4 more variables: local_politics <dbl>, sinclair2017 <dbl>,
## #   post <dbl>, month <ord>
```

`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

```r
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          0.134
## 2            0     1     0.0768         0.126
## 3            1     0     0.0936         0.137
## 4            1     1     0.0938         0.155
```

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              0.134          0.168
## 2            0     1   0.0768             0.126          0.167
## 3            1     0   0.0936             0.137          0.157
## 4            1     1   0.0938             0.155          0.139
```

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

```r
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

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

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           687
## 2 CBS           758
## 3 FOX           346
## 4 NBC           769
```

Helper function `count()`

`count()` does the same thing:

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

```
## # A tibble: 4 x 2
##   affiliation     n
##   <chr>       <int>
## 1 ABC           687
## 2 CBS           758
## 3 FOX           346
## 4 NBC           769
```

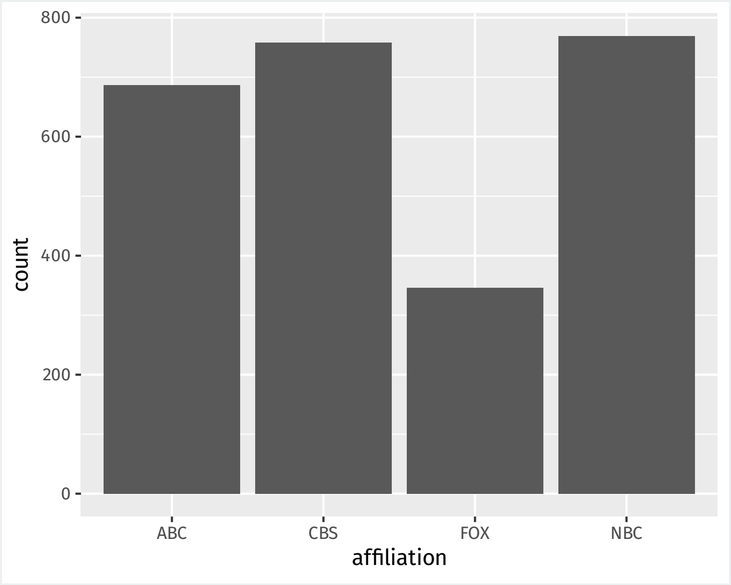
3/ Creating barplots

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.

```r
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.

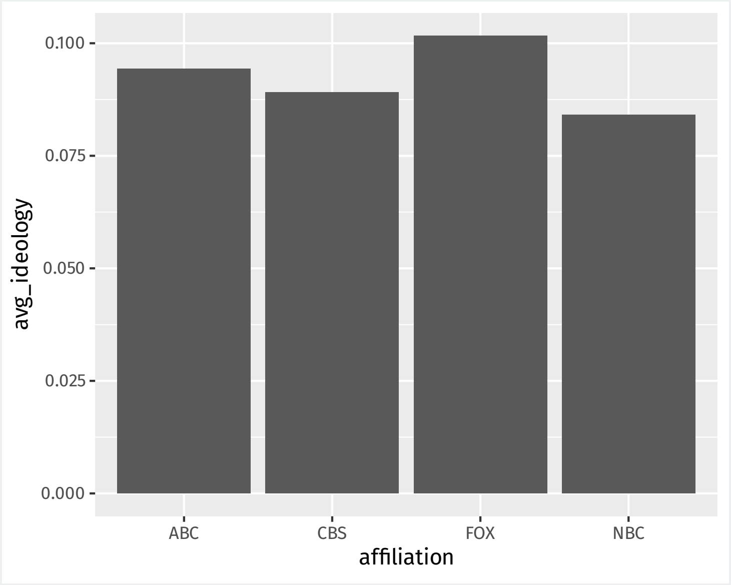
When the height of the bar is another variable in our data and not just a count, we set the x and y aesthetics and use `geom_col()` instead of `geom_bar()`.

Let's create a group summary:

```r
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
```

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



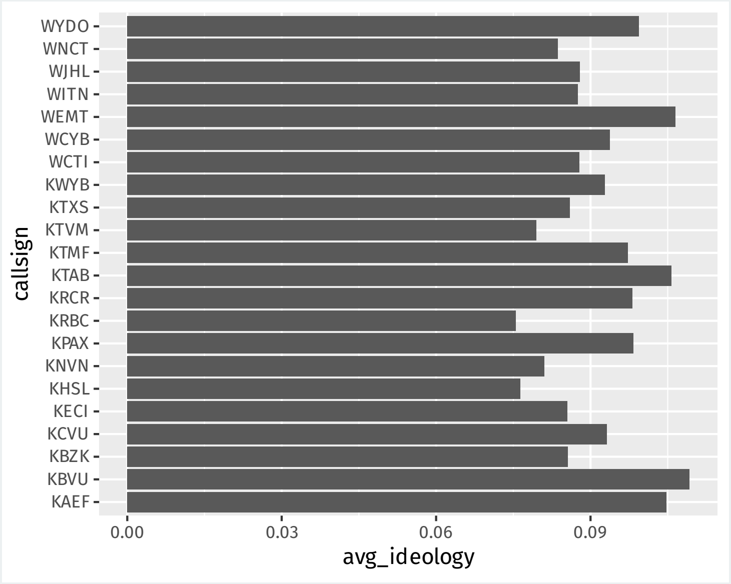
avg_ideology
0.100
0.075
0.050
0.025
0.000
ABC
CBS
FOX
NBC
affiliation

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)
```

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

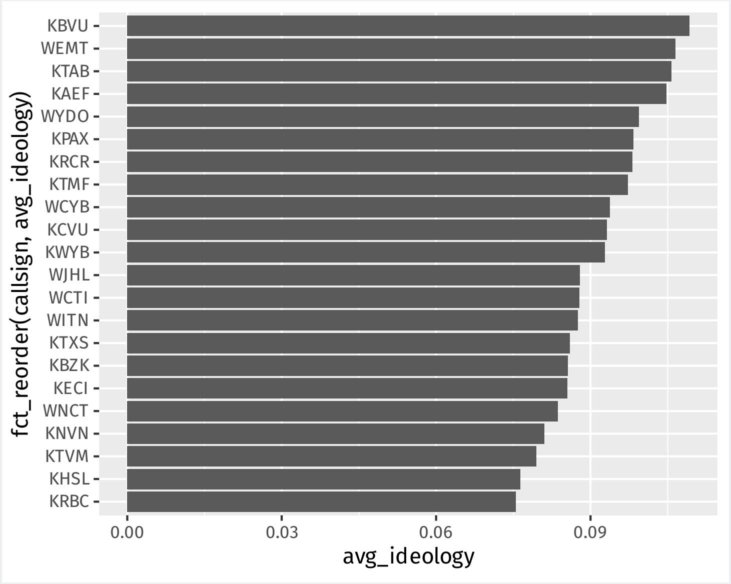


How do we reorder the stations?

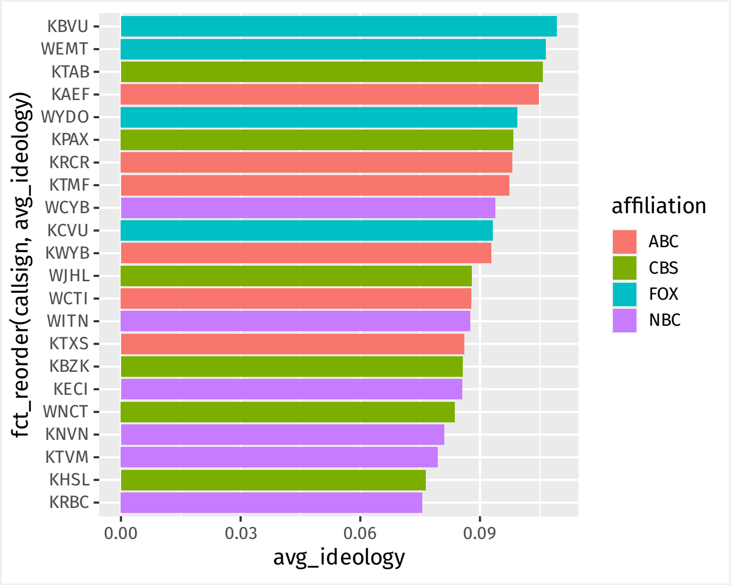
We would like to order the stations by ideology.

`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.

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



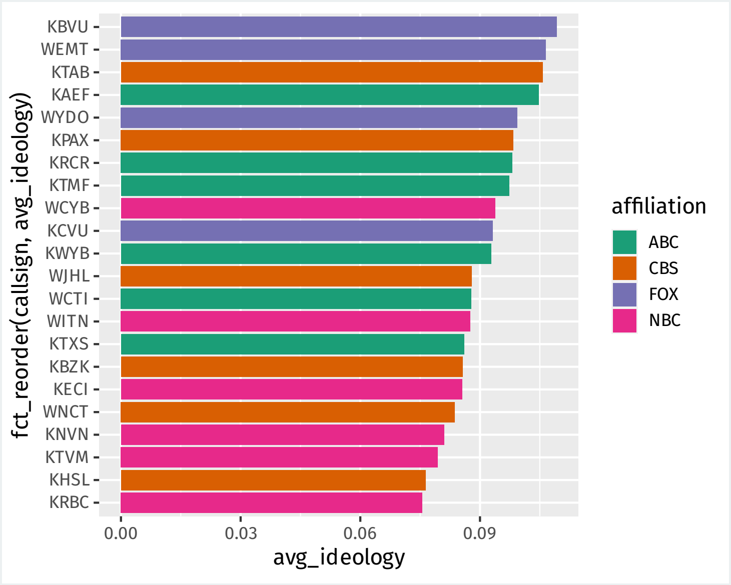
```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation))
```



Setting the color palette

We can use color palettes from a project called ColorBrewer

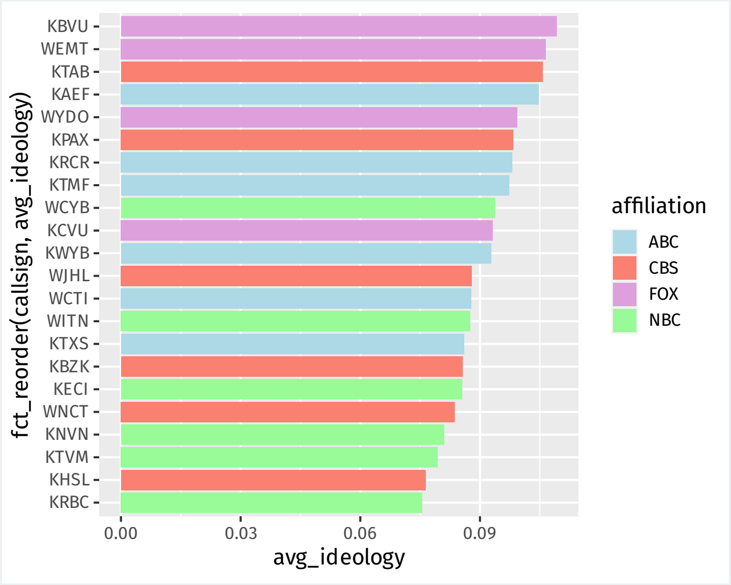
```
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_brewer(palette = "Dark2")
```



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

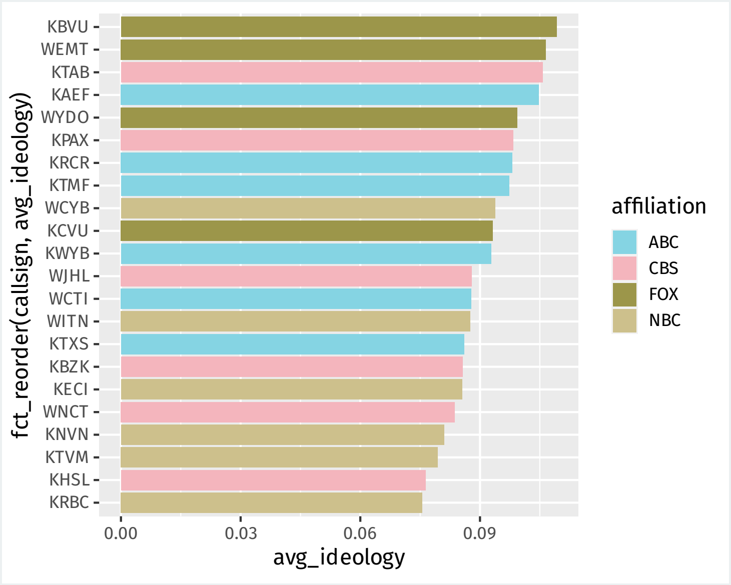
```r
ggplot(station_ideology,
       mapping = aes(x = avg_ideology,
                     y = fct_reorder(callsign, avg_ideology))) +
  geom_col(mapping = aes(fill = affiliation)) +
  scale_fill_manual(values = c(ABC = "lightblue",
                               CBS = "salmon",
                     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

```r
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