

Data Wrangling

Seung-Ho An, University of Arizona

- ① Review
- ② Data wrangling
- ③ Operating on rows
- ④ Operating on columns
- ⑤ Operating on groups
- ⑥ Creating a bar plot!

1. Review

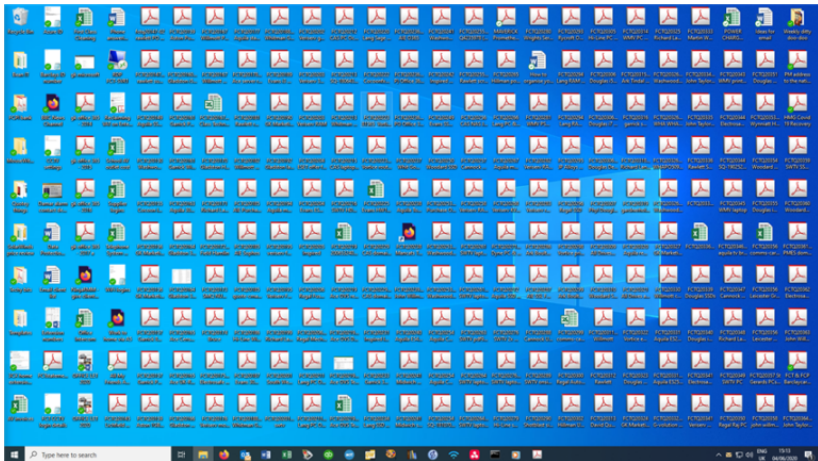
- Ways to look at data saved in data frames
 - `glimse()`

- Ways to look at data saved in data frames
 - `glimse()`
- A statistical graphic is a **m**apping of **d**ata variables to **a**esthetic attributes of **g**eometric objects.
- Visualizing data using `ggplot2` with a `geom` layer(s)
 - `geom_point` (scatterplot)
 - `geom_line` (linegraphs)
 - `geom_boxplot` (boxplots)
 - `geom_histogram` (histogram)
 - `geom_bar` (barplots)

2. Data wrangling

Let's talk about the **best** practices

Let's talk about the **best** practices



Certainly we don't want this..

Organize your folder first!

Create a project folder and name sub-folders

e.g.

```
\TPDstat  
  \Data  
  \Figure  
  \Assignment  
  \Table  
  \Figure  
  \TPD_is_the_Best
```

1. `setwd()`

- 1.1 Set your working directory as TPDstat (in the previous example)
- 1.2. load data from the subfolder, save figures and tables in the subfolders, and then generate the output using Rmarkdown!

1. `setwd()`

- 1.1 Set your working directory as TPDstat (in the previous example)
- 1.2. load data from the subfolder, save figures and tables in the subfolders, and then generate the output using Rmarkdown!

2. A (much) better way: create a R project in the folder and link it to git!

- 2.1. Click file -> New project -> designate a folder (e.g., TPDstat)
-> save the Rproject
- 2.2. Open the saved Rproject, begin using R (with a new R script(s), etc.)!
- 2.3. (optional) create a repository in Github and push/pull git (together with git, this is the best practice)!

R can certainly load almost all types of data; fwf (fixed width format), dcf, DIF, html, SPSS, SAS, Stata, Systat, XML, javascript, Excel, Minitab, etc.

An illustrated example: CSV

CSV is a form of a fixed width format; comma-separated values.
The CSV file would look like this:

```
Title,Author,ISBN13,Pages
1984,George Orwell,978-0451524935,268
Animal Farm,George Orwell,978-0451526342,144
Brave New World,Aldous Huxley,978-0060929879,288
Fahrenheit 451,Ray Bradbury,978-0345342966,208
Jane Eyre,Charlotte Brontë,978-0142437209,532
Wuthering Heights,Emily Brontë,978-0141439556,416
Agnes Grey,Anne Brontë,978-1593083236,256
Walden,Henry David Thoreau,978-1420922615,156
Walden Two,B. F. Skinner,978-0872207783,301
"Eats, Shoots & Leaves",Lynne Truss,978-1592400874,209
```

An illustrated example: CSV (cont)

To load the CSV file, we can use the `readr` package, which is pre-installed in the `tidyverse` package! Let's load then the `tidyverse` into our library to use `readr`. Let's load data on labor mobility in policing from Florida

```
## # A tibble: 88,446 x 13
##   Employ~1 payroll agency ptbid sepre~2 start~3 sep_d~4 Y_start Y_stop sex
##   <int> <int> <chr> <int> <chr> <chr> <chr> <int> <int> <chr>
## 1      22  96534 alach~ 65073 "VS-IA~ 12jul1~ "11apr~ 1990 2002 M
## 2     238 1424094 alach~ 85046 "Vol" 10dec1~ "11sep~ 1990 1996 M
## 3      97  533430 altam~ 233716 "Vol" 27mar2~ "01apr~ 2006 2016 M
## 4       1   4088 altha~ 88095 "TWA" 01jun2~ "09oct~ 2003 2003 M
## 5       7  30345 apala~ 243033 "Vol" 04jan2~ "18oct~ 2005 2005 M
## 6     98  536256 apopk~ 172716 "" 30sep2~ "" 2002 NA M
## 7     16  61232 arcad~ 281303 "" 01dec2~ "" 2008 2011 M
## 8      6  22440 astat~ 110808 "" 01apr2~ "" 2016 NA M
## 9     25 139954 atlan~ 230648 "" 18jul2~ "" 2011 NA M
## 10    12  67465 atlan~ 113978 "" 16feb2~ "" 2000 NA M
## # ... with 88,436 more rows, 3 more variables: race_code <chr>,
## #   birth_year <int>, education_level <chr>, and abbreviated variable names
## #   1: EmployeesNumber, 2: sepreason, 3: start_date, 4: sep_date, 5: sex_cod
```

You can use SQL in R using dplyr and/or sqldf packages!

```
library(dplyr)
# install.packages("sqldf")
library(sqldf)
```

Let's create a data.frame using built in iris data.

```
iris <- iris
```


Using dplyr:

```
iris |>  
  select(Sepal.Width) |>  
  filter(Sepal.Width>=4.0)
```

```
##   Sepal.Width  
## 1          4.0  
## 2          4.4  
## 3          4.1  
## 4          4.2
```

Using sqldf:

```
sqldf("select [Sepal.Width] from iris  
      where  
        [Sepal.Width]  >= 4.0")
```

```
##   Sepal.Width  
## 1          4.0  
## 2          4.4  
## 3          4.1  
## 4          4.2
```

Why data wrangling?

- The standard R object for dataset is the `data.frame`

- The standard R object for dataset is the `data.frame`
 - Each column is a vector of the same length

- The standard R object for dataset is the `data.frame`
 - Each column is a vector of the same length
 - Columns can be different types

- The standard R object for dataset is the `data.frame`
 - Each column is a vector of the same length
 - Columns can be different types
- Access columns with `$`: e.g., `mydata$myvariable`

```
options(width = 70)
mtcars$mpg
```

```
## [1] 21.0 21.0 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 17.8 16.4 17.3
## [14] 15.2 10.4 10.4 14.7 32.4 30.4 33.9 21.5 15.5 15.2 13.3 19.2 27.3
## [27] 26.0 30.4 15.8 19.7 15.0 21.4
```

Problems with data frames

mtcars

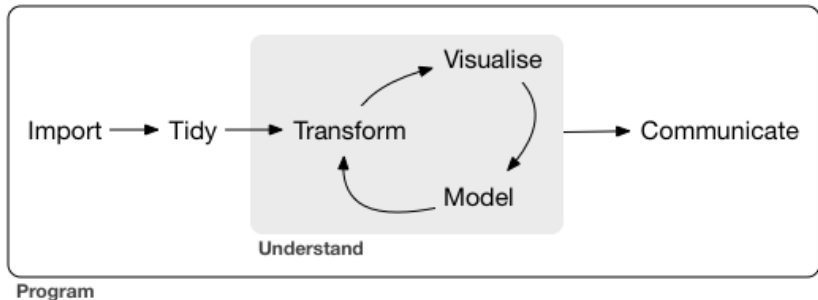
##	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear
## Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4
## Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4
## Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4
## Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3
## Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3
## Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3
## Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3
## Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4
## Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4
## Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4
## Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4
## Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3
## Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3
## Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3
## Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3
## Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3
## Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3
## Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4
## Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4
## Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4
## Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3
## Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3
## AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3

tibbles: a tidyverse alternative

```
library(ggplot2)
midwest
```

```
## # A tibble: 437 x 28      rows x columns
##   PID county state area popto~1 popde~2 popwh~3 popbl~4 popam~5      column types
##   <int> <chr>   <chr> <dbl> <int> <dbl> <int> <int> <int>
## 1  561 ADAMS   IL    0.052 66090 1271. 63917 1702 98
## 2  562 ALEXANDER IL    0.014 10626 759 7054 3496 19
## 3  563 BOND     IL    0.022 14991 681. 14477 429 35
## 4  564 BOONE    IL    0.017 30806 1812. 29344 127 46
## 5  565 BROWN    IL    0.018 5836 324. 5264 547 14
## 6  566 BUREAU   IL    0.05 35688 714. 35157 50 65
## 7  567 CALHOUN  IL    0.017 5322 313. 5298 1 8
## 8  568 CARROLL  IL    0.027 16805 622. 16519 111 30
## 9  569 CASS     IL    0.024 13437 560. 13384 16 8
## 10 570 CHAMPAIGN IL    0.058 173025 2983. 146506 16559 331
## # ... with 427 more rows, 19 more variables: popasian <int>,
## #   popother <int>, percwhite <dbl>, percblack <dbl>,
## #   percamerindian <dbl>, percasian <dbl>, percother <dbl>,
## #   popadults <int>, perchsd <dbl>, percollege <dbl>, percprof <dbl>,
## #   poppovertyknown <int>, percpovertyknown <dbl>,
## #   percbelowpoverty <dbl>, percchildbelowpovert <dbl>,
## #   percadultpoverty <dbl>, percelderlypoverty <dbl>, ... abridged output
```


Transform-Visualize-Model cycle



dplyr: a package for data transformation



- All `dplyr` functions:
 - Take a dataset as their first argument

dplyr: a package for data transformation



- All `dplyr` functions:
 - Take a dataset as their first argument
 - Manipulate the dataset in some way

dplyr: a package for data transformation



- All `dplyr` functions:
 - Take a dataset as their first argument
 - Manipulate the dataset in some way
 - Returns the manipulated dataset

Nested calls can be difficult to read (have to read inside out):

```
f(g(h(r(x))))
```

The pipe |> allows us to move output between functions (|> “and then”):

```
x |>  
  r() |>  
  h() |>  
  g() |>  
  h()
```

The piped output goes to the first argument by default

Application: Local news data

We will be using the `dplyr` package with local news data

Application: Local news data

We will be using the `dplyr` package with local news data

Martin and McCrain (2019) use data on local news at TV stations before and after a large acquisition by a conglomerate

Q: How does station ownership affect local news coverage?

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 local politics
sinclair2017	Station acquired by Sinclair group in Sept 2017
post	Date is before/after acquisition (0/1)
month	Month of the airdate

*# to load the TPDdata library, you should install it first using the codes
that I uploaded in the Teams chat*

```
library(TPDdata)
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
```

3. Operating on rows

filter() selects rows that satisfy the argument you pass it:

dplyr::filter() KEEP ROWS THAT satisfy your CONDITIONS

keep rows from... this data... ONLY IF... type is "otter" AND site is "bay"

```
filter(df, type == "otter" & site == "bay")
```

An orange, round, fuzzy cartoon character with a smiling face and small arms and legs. It is pointing its right hand towards a map. The map shows a green landmass with a blue bay area. Inside the bay, there is a small brown island with a house on it. The word 'BAY' is written in the water. A north arrow is in the bottom right corner of the map.

type	food	site
otter	urchin	bay
shark	seal	channel
otter	abalone	bay
otter	crab	wharf

A purple, round, fuzzy cartoon character with a smiling face and small arms and legs. It is pointing its right hand towards the 'channel' row of the table. Below it, a green, round, fuzzy cartoon character with a neutral face and small arms and legs is standing on a small green mound. It is pointing its right hand towards the 'wharf' row of the table. The table has orange highlights on the 'seal', 'channel', 'abalone', 'crab', and 'wharf' cells. There are checkmarks and X marks next to the rows: a checkmark next to the 'otter abalone bay' row and X marks next to the 'shark seal channel' and 'otter crab wharf' rows.

@allison_horst

Let's filter the local news data by observations aired on Tuesday (Tue) only; the variable name is `weekday`

How can we write the code?

Let's filter the local news data by observations aired on Tuesday (Tue) only; the variable name is weekday

How can we write the code?

```
# don't forget to call the library first (to use filter)  
library(dplyr)  
news |>  
  filter(weekday == "Tue")
```

```
news |>
  filter(weekday == "Tue")
```

```
## # A tibble: 626 x 10
##   calls~1 affil~2 date       weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>    <ord>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 KPAX     CBS      2017-06-06 Tue      NA        0.00357  0.194      0
## 2 KTAB     CBS      2017-06-06 Tue      NA        0.0945   0.109      0
## 3 KAEF     ABC      2017-06-13 Tue      0.0242   0.180     0.234     1
## 4 KBVU     FOX      2017-06-13 Tue      0.00894  0.186     0.245     1
## 5 KBZK     CBS      2017-06-13 Tue      0.129    0.306     0.0763    0
## 6 KCVU     FOX      2017-06-13 Tue      0.114    0.124     0.178     1
## 7 KECI     NBC      2017-06-13 Tue      0.115    0.283     0.0926    1
## 8 KHSL     CBS      2017-06-13 Tue      0.0821   0.274     0.248     0
## 9 KNVN     NBC      2017-06-13 Tue      0.120    0.261     0.253     0
## 10 KPAX    CBS      2017-06-13 Tue      0.0984   0.208     0.171     0
## # ... with 616 more rows, 2 more variables: post <dbl>, month <ord>,
## #   and abbreviated variable names 1: callsign, 2: affiliation,
## #   3: national_politics, 4: local_politics, 5: sinclair2017
```

Multiple conditions mean “and”

```
news |>  
  filter(weekday == "Tue",  
         affiliation == "FOX")
```

Multiple conditions mean “and”

```
news |>
  filter(weekday == "Tue",
         affiliation == "FOX")
```

```
## # A tibble: 130 x 10
##   calls~1 affil~2 date       weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>    <ord>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 KBVU    FOX      2017-06-13 Tue        0.00894  0.186    0.245     1
## 2 KCVU    FOX      2017-06-13 Tue        0.114    0.124    0.178     1
## 3 WEMT    FOX      2017-06-13 Tue        0.235    0.149    0.155     1
## 4 WYDO    FOX      2017-06-13 Tue        0.0949   0.182    0.180     1
## 5 KBVU    FOX      2017-06-20 Tue        NA        0.0229   0.268     1
## 6 KCVU    FOX      2017-06-20 Tue        NA        0.0170   0.261     1
## 7 KXVA    FOX      2017-06-20 Tue        NA        0.0203   0.0939    0
## 8 WEMT    FOX      2017-06-20 Tue        0.268    0.134    0.151     1
## 9 WYDO    FOX      2017-06-20 Tue        0.0590   0.155    0.0943    1
## 10 KBVU   FOX      2017-06-27 Tue        NA        0.0601   0.279     1
## # ... with 120 more rows, 2 more variables: post <dbl>, month <ord>,
## #   and abbreviated variable names 1: callsign, 2: affiliation,
## #   3: national_politics, 4: local_politics, 5: sinclair2017
```


Multiple conditions mean “and”

Then how about the codes below? Would we get the same results?

```
news |>
  filter(weekday == "Tue",
         affiliation == "Fox")
```

Multiple conditions mean “and”

Then how about the codes below? Would we get the same results?

```
news |>
  filter(weekday == "Tue",
         affiliation == "Fox")

## # A tibble: 0 x 10
## # ... with 10 variables: callsign <chr>, affiliation <chr>,
## #   date <date>, weekday <ord>, ideology <dbl>,
## #   national_politics <dbl>, local_politics <dbl>,
## #   sinclair2017 <dbl>, post <dbl>, month <ord>
```

Multiple conditions mean “and”

Then how about the codes below? Would we get the same results?

```
news |>
  filter(weekday == "Tue",
         affiliation == "Fox")

## # A tibble: 0 x 10
## # ... with 10 variables: callsign <chr>, affiliation <chr>,
## #   date <date>, weekday <ord>, ideology <dbl>,
## #   national_politics <dbl>, local_politics <dbl>,
## #   sinclair2017 <dbl>, post <dbl>, month <ord>
```

Wait... What..? What happened?

- Comparing two values/vectors:
 - > greater than
 - >= greater than or equal to

- Comparing two values/vectors:
 - $>$ greater than
 - \geq greater than or equal to
 - $<$ less than
 - \leq less than or equal to

- Comparing two values/vectors:
 - $>$ greater than
 - \geq greater than or equal to
 - $<$ less than
 - \leq less than or equal to
 - $==$ equal to
 - $!=$ not equal to

- Comparing two values/vectors:
 - > greater than
 - >= greater than or equal to
 - < less than
 - <= less than or equal to
 - == equal to
 - != not equal to
- Combining multiple logical statements:
 - & and
 - | or

Would this work?

```
news |>  
  filter(weekday = "Tue")
```


Would this work?

```
news |>  
  filter(weekday = "Tue")
```

```
## Error in `filter()`:  
## ! We detected a named input.  
## i This usually means that you've used `=` instead of `==`.  
## i Did you mean `weekday == "Tue"`?
```

Let's try a couple more filters!

```
news |>  
  filter(affiliation == "FOX" | affiliation == "ABC")
```

Let's try a couple more filters!

```
news |>
  filter(affiliation == "FOX" | affiliation == "ABC")

## # A tibble: 1,525 x 10
##   calls~1 affil~2 date       weekday ideology natio~3 local~4 sincl~5
##   <chr>    <chr>    <date>    <ord>      <dbl>    <dbl>    <dbl>    <dbl>
## 1 KXVA     FOX      2017-06-05 Mon        NA        0.0393  0.0262      0
## 2 KTMF     ABC      2017-06-07 Wed       8.42e-2  0.152    0.129      0
## 3 KTXS     ABC      2017-06-07 Wed      -4.88e-4  0.0925  0.0791      1
## 4 KXVA     FOX      2017-06-07 Wed        NA        0.00718  0.00479      0
## 5 KAEF     ABC      2017-06-08 Thu       4.26e-2  0.213    0.228      1
## 6 KBVU     FOX      2017-06-08 Thu      -8.60e-2  0.169    0.247      1
## 7 KTMF     ABC      2017-06-08 Thu       4.33e-2  0.179    0.139      0
## 8 KTXS     ABC      2017-06-08 Thu       6.27e-2  0.158    0.115      1
## 9 KXVA     FOX      2017-06-08 Thu        NA        0.0124  0.0873      0
## 10 WCTI    ABC      2017-06-08 Thu       1.39e-1  0.225    0.0759      1
## # ... with 1,515 more rows, 2 more variables: post <dbl>,
## #   month <ord>, and abbreviated variable names 1: callsign,
## #   2: affiliation, 3: national_politics, 4: local_politics,
## #   5: sinclair2017
```

When combining | and ==, useful to use %in%:

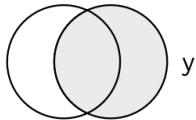
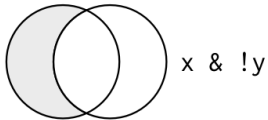
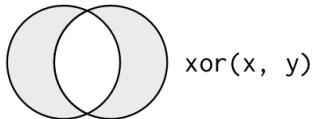
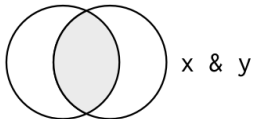
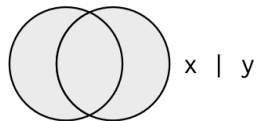
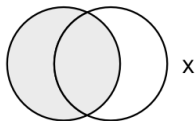
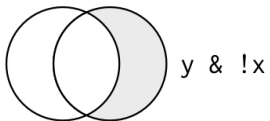
```
news |>
  filter(weekday %in% c("Mon", "Fri"))
```

A tibble: 1,253 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	KAEF	ABC	2017-06-09	Fri	0.0870	0.153	0.269	1
## 5	KBVU	FOX	2017-06-09	Fri	NA	0.0553	0.384	1
## 6	KECI	NBC	2017-06-09	Fri	0.115	0.216	0.108	1
## 7	KPAX	CBS	2017-06-09	Fri	0.0882	0.315	0.128	0
## 8	KRBC	NBC	2017-06-09	Fri	0.0929	0.152	0.147	0
## 9	KTAB	CBS	2017-06-09	Fri	0.0588	0.0711	0.176	0
## 10	KTMF	ABC	2017-06-09	Fri	NA	0.0495	0.0999	0

... with 1,243 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

Complicated logicals



`arrange()` will reorder the rows based on the values of the columns (the default option is ascending)

With multiple arguments, sort by first argument, then second, then third...

arrange by callsign then date

```
news |>
  arrange(callsign, date)
```

```
## # 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 KAEF      ABC     2017-06-08 Thu      0.0426  0.213    0.228      1
## 2 KAEF      ABC     2017-06-09 Fri      0.0870  0.153    0.269      1
## 3 KAEF      ABC     2017-06-12 Mon      0.0135  0.149    0.188      1
## 4 KAEF      ABC     2017-06-13 Tue      0.0242  0.180    0.234      1
## 5 KAEF      ABC     2017-06-14 Wed      0.123   0.182    0.0968     1
## 6 KAEF      ABC     2017-06-15 Thu      0.0778  0.114    0.203      1
## 7 KAEF      ABC     2017-06-16 Fri      NA      0.109    0.176      1
## 8 KAEF      ABC     2017-06-19 Mon      0.778   0.0823  0.179      1
## 9 KAEF      ABC     2017-06-20 Tue      0.115   0.131    0.163      1
## 10 KAEF     ABC     2017-06-21 Wed     -0.315   0.130    0.192      1
## # ... 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
```

Which station-dates were the most liberal?

```
news |>
  arrange(ideology)
```

```
## # 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     2017-10-19 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
## 6 KAEF      ABC     2017-06-21 Wed      -0.315    0.130     0.192      1
## 7 KTMF      ABC     2017-12-01 Fri      -0.303    0.179     0.150      0
## 8 KWYB      ABC     2017-12-01 Fri      -0.303    0.160     0.161      0
## 9 KTVM      NBC     2017-09-01 Fri      -0.302    0.0507    0.106      1
## 10 KNVN     NBC     2017-12-08 Fri      -0.299    0.121     0.211      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
```


Which station-dates were the most conservative?

```
news |>
  arrange(desc(ideology))
```

```
## # 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 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
## 6 KTMF      ABC     2017-11-06 Mon      0.455  0.138   0.154      0
## 7 KAEF      ABC     2017-06-29 Thu      0.447  0.126   0.220      1
## 8 KPAX      CBS     2017-11-23 Thu      0.437  0.125   0.128      0
## 9 KTAB      CBS     2017-11-16 Thu      0.427  0.0631  0.104      0
## 10 KCVU     FOX     2017-07-06 Thu      0.406  0.154   0.148      1
## # ... 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
```

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

Let's look at the first and fifth observations

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

Let's look at the first and fifth observations

```
## first and third row
```

```
news |>
  slice(1, 5)
```

```
## # 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 KTAB      CBS      2017-06-06 Tue        NA    0.0945  0.109      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:

```
## # 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      2017-06-05 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
## # ... 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(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(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
```

4. Operating on columns

`select()` selects columns via their names

Selecting based on names

```
news |>
  select(callsign, date, ideology)
```

```
## # A tibble: 3,137 x 3
##   callsign date      ideology
##   <chr>    <date>      <dbl>
## 1 KRBC    2017-06-05    NA
## 2 KTAB    2017-06-05    NA
## 3 KXVA    2017-06-05    NA
## 4 KPAX    2017-06-06    NA
## 5 KTAB    2017-06-06    NA
## 6 KECI    2017-06-07    0.0655
## 7 KPAX    2017-06-07    0.0853
## 8 KRBC    2017-06-07    0.0183
## 9 KTAB    2017-06-07    0.0850
## 10 KTMF    2017-06-07    0.0842
## # ... with 3,127 more rows
```

Selecting based on a range of variables

```
news |>
  select(callsign:ideology)
```

```
## # A tibble: 3,137 x 5
##   callsign affiliation date       weekday ideology
##   <chr>      <chr>      <date>      <ord>      <dbl>
## 1 KRBC      NBC        2017-06-05 Mon         NA
## 2 KTAB      CBS        2017-06-05 Mon         NA
## 3 KXVA      FOX        2017-06-05 Mon         NA
## 4 KPAX      CBS        2017-06-06 Tue         NA
## 5 KTAB      CBS        2017-06-06 Tue         NA
## 6 KECI      NBC        2017-06-07 Wed         0.0655
## 7 KPAX      CBS        2017-06-07 Wed         0.0853
## 8 KRBC      NBC        2017-06-07 Wed         0.0183
## 9 KTAB      CBS        2017-06-07 Wed         0.0850
## 10 KTMF     ABC        2017-06-07 Wed         0.0842
## # ... with 3,127 more rows
```

What would this code do?

```
news |>  
  select(!callsign:ideology)
```

What would this code do?

```
news |>
  select(!callsign:ideology)
```

Selecting all not in a range

```
## # A tibble: 3,137 x 5
##   national_politics local_politics sinclair2017 post month
##           <dbl>           <dbl>         <dbl> <dbl> <ord>
## 1           0.0286           0.0190             0     0 Jun
## 2           0.0286           0.0190             0     0 Jun
## 3           0.0393           0.0262             0     0 Jun
## 4           0.00357          0.194             0     0 Jun
## 5           0.0945           0.109             0     0 Jun
## 6           0.225           0.148             1     0 Jun
## 7           0.283           0.123             0     0 Jun
## 8           0.130           0.189             0     0 Jun
## 9           0.0901           0.138             0     0 Jun
## 10          0.152           0.129             0     0 Jun
## # ... with 3,127 more rows
```

Selecting all numeric columns

```
news |>
  select(where(is.numeric))
```

```
## # A tibble: 3,137 x 5
##   ideology national_politics local_politics sinclair2017 post
##   <dbl>         <dbl>         <dbl>         <dbl> <dbl>
## 1 NA           0.0286         0.0190         0      0
## 2 NA           0.0286         0.0190         0      0
## 3 NA           0.0393         0.0262         0      0
## 4 NA           0.00357        0.194         0      0
## 5 NA           0.0945         0.109         0      0
## 6 0.0655        0.225         0.148         1      0
## 7 0.0853        0.283         0.123         0      0
## 8 0.0183        0.130         0.189         0      0
## 9 0.0850        0.0901        0.138         0      0
## 10 0.0842        0.152         0.129         0      0
## # ... with 3,127 more rows
```

Combining multiple selections

```
news |>
  select(callsign:weekday, ends_with("politics"))
```

```
## # A tibble: 3,137 x 6
##   callsign affiliation date      weekday national_politics local_p~1
##   <chr>      <chr>      <date>      <ord>          <dbl>      <dbl>
## 1 KRBC      NBC        2017-06-05 Mon           0.0286      0.0190
## 2 KTAB      CBS        2017-06-05 Mon           0.0286      0.0190
## 3 KXVA      FOX        2017-06-05 Mon           0.0393      0.0262
## 4 KPAX      CBS        2017-06-06 Tue           0.00357     0.194
## 5 KTAB      CBS        2017-06-06 Tue           0.0945     0.109
## 6 KECI      NBC        2017-06-07 Wed           0.225      0.148
## 7 KPAX      CBS        2017-06-07 Wed           0.283      0.123
## 8 KRBC      NBC        2017-06-07 Wed           0.130      0.189
## 9 KTAB      CBS        2017-06-07 Wed           0.0901     0.138
## 10 KTMF     ABC        2017-06-07 Wed           0.152      0.129
## # ... with 3,127 more rows, and abbreviated variable name
## #   1: local_politics
```

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

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


```
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_poli~1 nation~2 natio~3
##   <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, and abbreviated variable names
## #   1: local_politics, 2: national_local_diff,
## #   3: national_politics_perc
```

`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

What would the codes below do?

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

What would the codes below do?

```
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
```

5. Operating on groups

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

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

ideological slant by month

```
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)
  )
```

`summarise()` has grouped output by 'sinclair2017'. You
using the `.groups` argument.

```
## # 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)
  )
```

```
## Warning: There was 1 warning in `summarize()`.
## i In argument: `across(where(is.numeric), mean, na.rm = TRUE)`.
## i In group 1: `sinclair2017 = 0`, `post = 0`.
## Caused by warning:
## ! The `...` argument of `across()` is deprecated as of dplyr 1.1.0.
## Supply arguments directly to `.fns` through an anonymous function
## instead.
##
## # Previously
##   across(a:b, mean, na.rm = TRUE)
##
## # Now
##   across(a:b, \(x) mean(x, na.rm = TRUE))

## `summarise()` has grouped output by 'sinclair2017'. You can override
## using the `.groups` argument.

## # 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
```

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

`summarise()` has grouped output by 'sinclair2017'. You can override
using the `.groups` argument.

```
## # 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
```

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

`summarise()` has grouped output by 'sinclair2017'. You can override
using the `.groups` argument.

```
## # 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	0.0785518
Jul	0.1032917
Aug	0.1049908
Sep	0.0751067
Oct	0.0861639
Nov	0.0971796
Dec	0.0773873

Nice table! But, column names?

Producing 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	0.0785518
Jul	0.1032917
Aug	0.1049908
Sep	0.0751067
Oct	0.0861639
Nov	0.0971796
Dec	0.0773873

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
```

`count()` does the same thing:

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



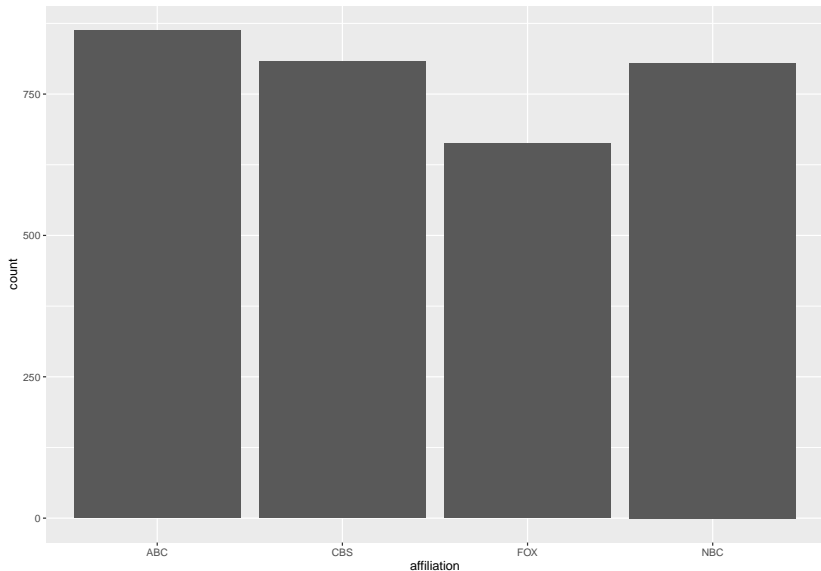
```
## # A tibble: 4 x 2
##   affiliation      n
##   <chr>         <int>
## 1 ABC           863
## 2 CBS           807
## 3 FOX           662
## 4 NBC           805
```

6. Creating barplots

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

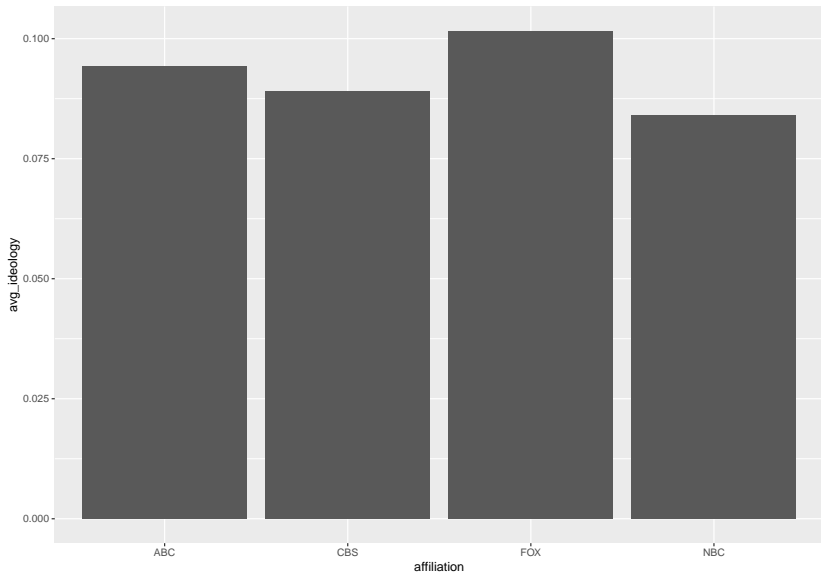
When the height of the bar is another variable in our data and not just a count, we set the x and y aesthetics and use `geom_col()` instead of `geom_bar()`

Let's create a group summary: the average ideology scores by affiliation

```
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()
```



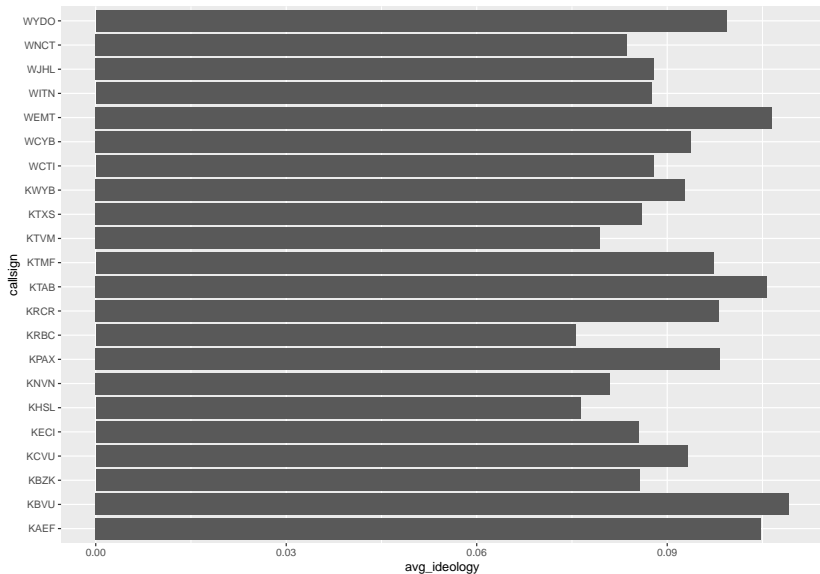
A more complicated example

Let's create a bar plot 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)
```

na.omit() helps removing NA

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

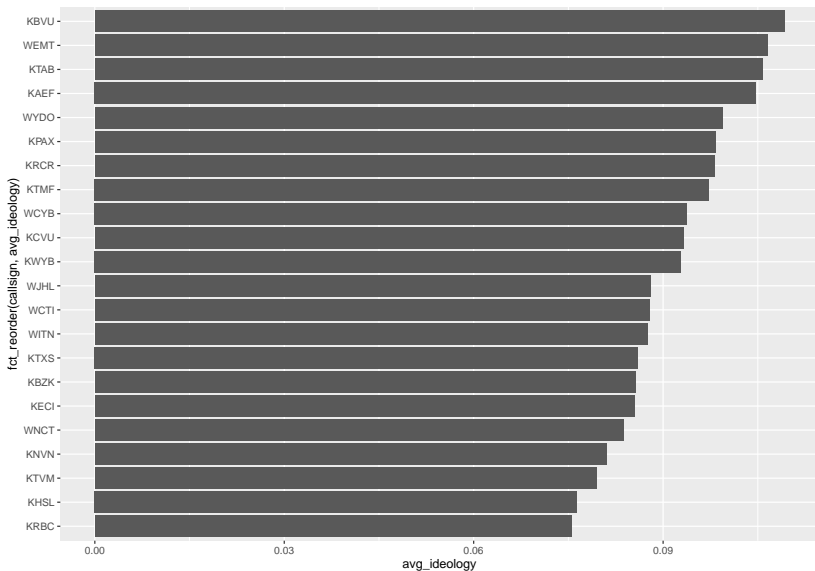


How do we reorder the stations?

We would like to order the stations by ideology

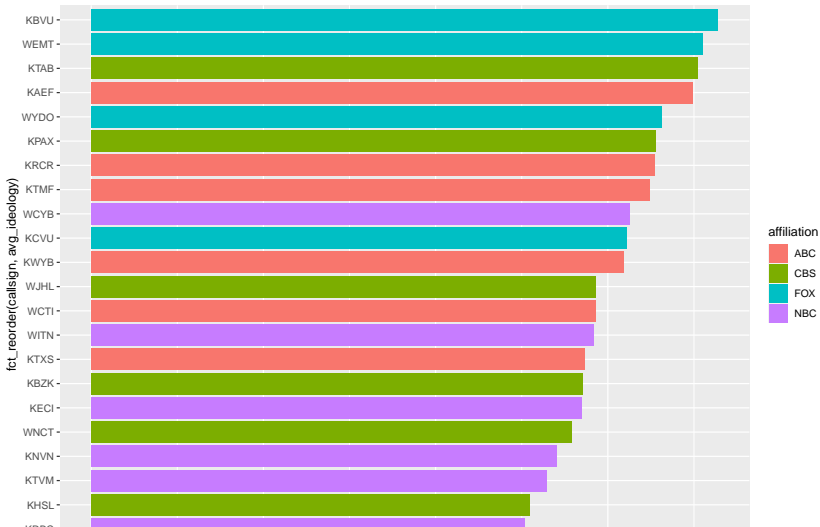
`fct_reorder(group, order_var)` function (loaded with `tidyverse`) will reorder the groups by the order bar (low to high).
Easiest to put this in the mapping.

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



Adding color by affiliation

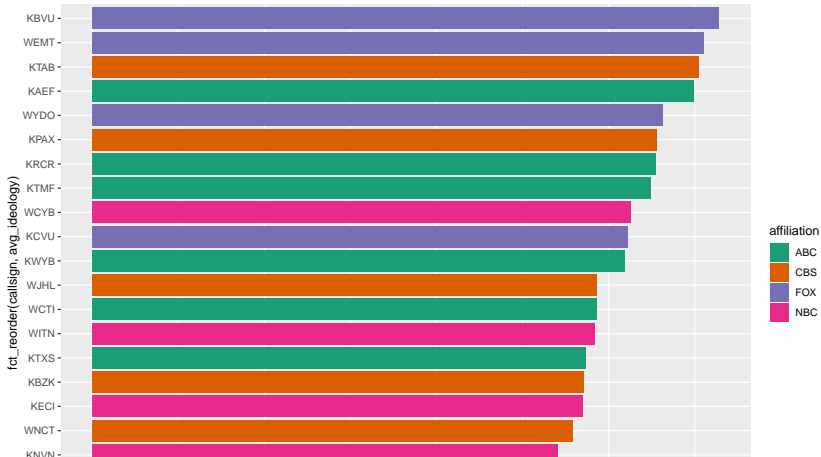
```
ggplot(na.omit(station_ideology),  
       mapping = aes(x = avg_ideology,  
                     y = fct_reorder(callsign, avg_ideology))) +  
  geom_col(mapping = aes(fill = affiliation))
```



Setting the color palette

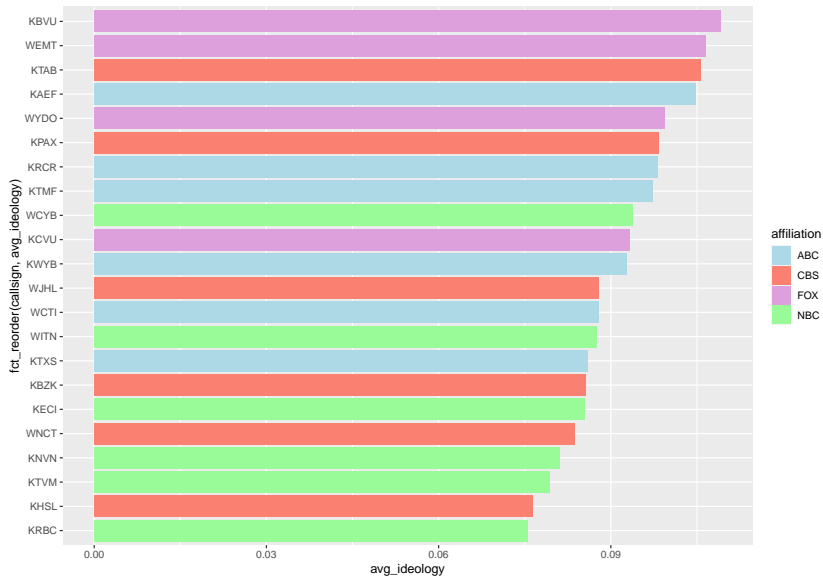
We can use color palettes from a project called ColorBrewer

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



Manually setting the color palette

```
ggplot(na.omit(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"))
```



Other packages provide more palettes:

```
library(wesanderson)
ggplot(na.omit(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"))
```

