Manipulating Tables with dplyr (contd)

Session 3

Kieran Healy

Statistical Horizons, September 2021

Window functions and moving averages

Load our libraries

```
library(here)      # manage file paths
```

```
## here() starts at /Users/kjhealy/Documents/courses/data_wrangling
```

```
library(socviz)    # data and some useful functions
```

```
##
## Attaching package: 'socviz'
```

```
## The following object is masked from 'package:kjhutils':
##
##     %nin%
```

```
library(tidyverse) # your friend and mine
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.4     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   2.0.1     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x readr::edition_get()   masks testthat::edition_get()
## x dplyr::filter()        masks stats::filter()
## x purrr::is_null()       masks testthat::is_null()
## x dplyr::lag()           masks stats::lag()
## x readr::local_edition() masks testthat::local_edition()
## x dplyr::matches()       masks tidyr::matches(), testthat::matches()
```

dplyr's window functions

Ranking and cumulation within groups.

```
## Data on COVID-19
library(covdata)

covnat_weekly
```

```
## # A tibble: 12,720 × 11
##    date       year_week cname       iso3       pop cases deaths cu_cases cu_deaths
##    <date>     <chr>     <chr>       <chr>    <dbl> <dbl>  <dbl>    <dbl>     <dbl>
##  1 2019-12-30 2020-01   Afghanistan AFG   38928341     0      0        0         0
##  2 2020-01-06 2020-02   Afghanistan AFG   38928341     0      0        0         0
##  3 2020-01-13 2020-03   Afghanistan AFG   38928341     0      0        0         0
##  4 2020-01-20 2020-04   Afghanistan AFG   38928341     0      0        0         0
##  5 2020-01-27 2020-05   Afghanistan AFG   38928341     0      0        0         0
##  6 2020-02-03 2020-06   Afghanistan AFG   38928341     0      0        0         0
##  7 2020-02-10 2020-07   Afghanistan AFG   38928341     0      0        0         0
##  8 2020-02-17 2020-08   Afghanistan AFG   38928341     0      0        0         0
##  9 2020-02-24 2020-09   Afghanistan AFG   38928341     1      0        1         0
## 10 2020-03-02 2020-10   Afghanistan AFG   38928341     3      0        4         0
## # … with 12,710 more rows, and 2 more variables: r14_cases <dbl>,
## #   r14_deaths <dbl>
```

dplyr's window functions

`cumsum()` gives cumulative sums

```
covnat_weekly %>%
  filter(cname == "United States") %>%
  select(date, cname, iso3, cases) %>%
  mutate(cumulative = cumsum(cases))
```

```
## # A tibble: 0 × 5
## # … with 5 variables: date <date>, cname <chr>, iso3 <chr>, cases <dbl>,
## #   cumulative <dbl>
```

dplyr's window functions

`cume_dist()` gives the proportion of values less than or equal to the current value.

```
covnat_weekly %>%
  select(date, cname, iso3, deaths) %>%
  filter(cname == "United States") %>%
  filter(cume_dist(desc(deaths)) < 0.1) # i.e. Top 10%
```

```
## # A tibble: 0 × 4
## # … with 4 variables: date <date>, cname <chr>, iso3 <chr>, deaths <dbl>
```

The `dplyr` vignette on Window functions is good.

An application

```
covus %>%
  filter(measure == "death") %>%
  group_by(state) %>%
  arrange(state, desc(date)) %>%
  filter(state %in% "NY")
```

```
## # A tibble: 371 × 7
## # Groups:   state [1]
##    date       state fips  data_quality_grade measure count measure_label
##    <date>     <chr> <chr> <lgl>              <chr>   <dbl> <chr>
##  1 2021-03-07 NY    36    NA                 death   39029 Deaths
##  2 2021-03-06 NY    36    NA                 death   38970 Deaths
##  3 2021-03-05 NY    36    NA                 death   38891 Deaths
##  4 2021-03-04 NY    36    NA                 death   38796 Deaths
##  5 2021-03-03 NY    36    NA                 death   38735 Deaths
##  6 2021-03-02 NY    36    NA                 death   38660 Deaths
##  7 2021-03-01 NY    36    NA                 death   38577 Deaths
##  8 2021-02-28 NY    36    NA                 death   38497 Deaths
##  9 2021-02-27 NY    36    NA                 death   38407 Deaths
## 10 2021-02-26 NY    36    NA                 death   38321 Deaths
## # … with 361 more rows
```

Here the `count` measure is *cumulative* deaths. What if we want to recover the daily count for all the states in the data?

An application

`dplyr` has `lead()` and `lag()` functions. These allow you to access the previous and next values in a vector. You can calculate offsets this way.

```
my_vec <- c(1:20)
my_vec
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
lag(my_vec) # first element has no lag
```

```
##  [1] NA  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19
```

```
my_vec - lag(my_vec)
```

```
##  [1] NA  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
```

An application

We can write the expression directly:

```
covus %>%
  select(-data_quality_grade) %>%
  filter(measure == "death") %>%
  group_by(state) %>%
  arrange(date) %>%
  mutate(deaths_daily = count - lag(count, order_by = date)) %>%
  arrange(state, desc(date)) %>%
  filter(state %in% "NY")
```

```
## # A tibble: 371 × 7
## # Groups:   state [1]
##    date       state fips  measure count measure_label deaths_daily
##    <date>     <chr> <chr> <chr>   <dbl> <chr>                <dbl>
##  1 2021-03-07 NY    36    death   39029 Deaths                  59
##  2 2021-03-06 NY    36    death   38970 Deaths                  79
##  3 2021-03-05 NY    36    death   38891 Deaths                  95
##  4 2021-03-04 NY    36    death   38796 Deaths                  61
##  5 2021-03-03 NY    36    death   38735 Deaths                  75
##  6 2021-03-02 NY    36    death   38660 Deaths                  83
##  7 2021-03-01 NY    36    death   38577 Deaths                  80
##  8 2021-02-28 NY    36    death   38497 Deaths                  90
##  9 2021-02-27 NY    36    death   38407 Deaths                  86
## 10 2021-02-26 NY    36    death   38321 Deaths                  94
## # … with 361 more rows
```

Writing our own functions

But we could also write a function to do this.

We write functions using the special `function()` function.*

```
my_fun <- function(x) {
  x + 1
}

my_fun # we've created the function; it's just an object
```

```
## function(x) {
##   x + 1
## }
```

```
my_fun(x = 1) # But we can supply it with an input!
```

```
## [1] 2
```

```
my_fun(10)
```

```
## [1] 11
```

*Nerds love this sort of stuff.

Writing our own functions

We write our function. It's just the expression we originally wrote, wrapped up.

```
get_daily_count <- function(count, date){
  count - lag(count, order_by = date)
}
```

This function has no generality, error-handling, or anything else. It's a once-off.

Writing our own functions

Now we can use it like any other:

```
covus %>%
  filter(measure == "death") %>%
  select(-data_quality_grade) %>%
  group_by(state) %>%
  arrange(date) %>%
  mutate(deaths_daily = get_daily_count(count, date)) %>%
  arrange(state, desc(date)) %>%
  filter(state %in% "NY")
```

```
## # A tibble: 371 × 7
## # Groups:   state [1]
##    date       state fips  measure count measure_label deaths_daily
##    <date>     <chr> <chr> <chr>   <dbl> <chr>                <dbl>
##  1 2021-03-07 NY    36    death   39029 Deaths                  59
##  2 2021-03-06 NY    36    death   38970 Deaths                  79
##  3 2021-03-05 NY    36    death   38891 Deaths                  95
##  4 2021-03-04 NY    36    death   38796 Deaths                  61
##  5 2021-03-03 NY    36    death   38735 Deaths                  75
##  6 2021-03-02 NY    36    death   38660 Deaths                  83
##  7 2021-03-01 NY    36    death   38577 Deaths                  80
##  8 2021-02-28 NY    36    death   38497 Deaths                  90
##  9 2021-02-27 NY    36    death   38407 Deaths                  86
## 10 2021-02-26 NY    36    death   38321 Deaths                  94
## # … with 361 more rows
```

Not super-useful quite yet, but if our task had more steps ...

Tidy moving averages with `slider`

`dplyr`'s window functions don't include moving averages.

There are several options, notably `RcppRoll`

We'll use the `slider` package.

```
# install.packages("slider")
library(slider)
```

Tidy moving averages with `slider`

```
covus %>%
  filter(measure == "death") %>%
  select(-data_quality_grade) %>%
  group_by(state) %>%
  arrange(date) %>%
  mutate(
    deaths_daily = get_daily_count(count, date),
    deaths7 = slide_mean(deaths_daily,
                         before = 7,
                         na_rm = TRUE)) %>%
  arrange(state, desc(date)) %>%
  filter(state %in% "NY")
```

```
## # A tibble: 371 × 8
## # Groups:   state [1]
##    date       state fips  measure count measure_label deaths_daily deaths7
##    <date>     <chr> <chr> <chr>   <dbl> <chr>                <dbl>   <dbl>
##  1 2021-03-07 NY    36    death   39029 Deaths                  59    77.8
##  2 2021-03-06 NY    36    death   38970 Deaths                  79    81.1
##  3 2021-03-05 NY    36    death   38891 Deaths                  95    83
##  4 2021-03-04 NY    36    death   38796 Deaths                  61    82.6
##  5 2021-03-03 NY    36    death   38735 Deaths                  75    88
##  6 2021-03-02 NY    36    death   38660 Deaths                  83    89.9
##  7 2021-03-01 NY    36    death   38577 Deaths                  80    90.8
##  8 2021-02-28 NY    36    death   38497 Deaths                  90    90.1
##  9 2021-02-27 NY    36    death   38407 Deaths                  86    91.5
## 10 2021-02-26 NY    36    death   38321 Deaths                  94    95.6
## # … with 361 more rows
```

Tidy moving averages with `slider`

```
    deaths7 = slide_mean(deaths_daily, #<<
                         before = 7, #<<
                         na_rm = TRUE)) %>% #<<
```

Notice the Tidyverse-style `na_rm` argument rather than the usual base `na.rm`

The package provides a lot of different functions, from general-purpose `slide_max()`, `slide_min()` to more specialized sliding functions. In particular note e.g. `slide_index_mean()` that addresses some subtleties in averaging over dates with gaps.

Tidying up after yourself with `relocate()`

```
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 …
## # … with 2,857 more rows, and 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>
```

Shuffle columns around

```
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 …
## # … with 2,857 more rows, and 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>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income"))
```

```
## # A tibble: 2,867 × 19
##    region   bigregion  year    id ballot    age childs sibs  degree  race  sex
##    <fct>    <fct>     <dbl> <dbl> <label> <dbl>  <dbl> <lab> <fct>   <fct> <fct>
##  1 New Eng… Northeast  2016     1 1          47      3 2     Bachel… White Male
##  2 New Eng… Northeast  2016     2 2          61      0 3     High S… White Male
##  3 New Eng… Northeast  2016     3 3          72      2 3     Bachel… White Male
##  4 New Eng… Northeast  2016     4 1          43      4 3     High S… White Fema…
##  5 New Eng… Northeast  2016     5 3          55      2 2     Gradua… White Fema…
##  6 New Eng… Northeast  2016     6 2          53      2 2     Junior… White Fema…
##  7 New Eng… Northeast  2016     7 1          50      2 2     High S… White Male
##  8 Middle … Northeast  2016     8 3          23      3 6     High S… Other Fema…
##  9 Middle … Northeast  2016     9 1          45      3 5     High S… Black Male
## 10 Middle … Northeast  2016    10 3          71      4 1     Junior… White Male
## # … with 2,857 more rows, and 8 more variables: padeg <fct>, partyid <fct>,
## #   polviews <fct>, partners <fct>, pres12 <labelled>, partners_rc <fct>,
## #   income16 <fct>, income_rc <fct>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income")) %>%
  rename(children = childs,
         siblings = sibs)
```

```
## # A tibble: 2,867 × 19
##    region   bigregion  year    id ballot    age children siblings  degree  race
##    <fct>    <fct>     <dbl> <dbl> <label> <dbl>    <dbl> <labelle> <fct>   <fct>
##  1 New Eng… Northeast  2016     1 1          47        3 2         Bachel… White
##  2 New Eng… Northeast  2016     2 2          61        0 3         High S… White
##  3 New Eng… Northeast  2016     3 3          72        2 3         Bachel… White
##  4 New Eng… Northeast  2016     4 1          43        4 3         High S… White
##  5 New Eng… Northeast  2016     5 3          55        2 2         Gradua… White
##  6 New Eng… Northeast  2016     6 2          53        2 2         Junior… White
##  7 New Eng… Northeast  2016     7 1          50        2 2         High S… White
##  8 Middle … Northeast  2016     8 3          23        3 6         High S… Other
##  9 Middle … Northeast  2016     9 1          45        3 5         High S… Black
## 10 Middle … Northeast  2016    10 3          71        4 1         Junior… White
## # … with 2,857 more rows, and 9 more variables: sex <fct>, padeg <fct>,
## #   partyid <fct>, polviews <fct>, partners <fct>, pres12 <labelled>,
## #   partners_rc <fct>, income16 <fct>, income_rc <fct>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income")) %>%
  rename(children = childs,
         siblings = sibs) %>%
  relocate(id)
```

```
## # A tibble: 2,867 × 19
##       id region   bigregion  year ballot    age children siblings  degree  race
##    <dbl> <fct>    <fct>     <dbl> <label> <dbl>    <dbl> <labelle> <fct>   <fct>
##  1     1 New Eng… Northeast  2016 1          47        3 2         Bachel… White
##  2     2 New Eng… Northeast  2016 2          61        0 3         High S… White
##  3     3 New Eng… Northeast  2016 3          72        2 3         Bachel… White
##  4     4 New Eng… Northeast  2016 1          43        4 3         High S… White
##  5     5 New Eng… Northeast  2016 3          55        2 2         Gradua… White
##  6     6 New Eng… Northeast  2016 2          53        2 2         Junior… White
##  7     7 New Eng… Northeast  2016 1          50        2 2         High S… White
##  8     8 Middle … Northeast  2016 3          23        3 6         High S… Other
##  9     9 Middle … Northeast  2016 1          45        3 5         High S… Black
## 10    10 Middle … Northeast  2016 3          71        4 1         Junior… White
## # … with 2,857 more rows, and 9 more variables: sex <fct>, padeg <fct>,
## #   partyid <fct>, polviews <fct>, partners <fct>, pres12 <labelled>,
## #   partners_rc <fct>, income16 <fct>, income_rc <fct>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income")) %>%
  rename(children = childs,
         siblings = sibs) %>%
  relocate(id) %>%
  select(-ballot)
```

```
## # A tibble: 2,867 × 18
##       id region bigregion  year   age children siblings degree race  sex   padeg
##    <dbl> <fct>  <fct>     <dbl> <dbl>    <dbl> <labell> <fct>  <fct> <fct> <fct>
##  1     1 New E… Northeast  2016    47        3 2        Bache… White Male  Grad…
##  2     2 New E… Northeast  2016    61        0 3        High … White Male  Lt H…
##  3     3 New E… Northeast  2016    72        2 3        Bache… White Male  High…
##  4     4 New E… Northeast  2016    43        4 3        High … White Fema… <NA>
##  5     5 New E… Northeast  2016    55        2 2        Gradu… White Fema… Bach…
##  6     6 New E… Northeast  2016    53        2 2        Junio… White Fema… <NA>
##  7     7 New E… Northeast  2016    50        2 2        High … White Male  High…
##  8     8 Middl… Northeast  2016    23        3 6        High … Other Fema… Lt H…
##  9     9 Middl… Northeast  2016    45        3 5        High … Black Male  Lt H…
## 10    10 Middl… Northeast  2016    71        4 1        Junio… White Male  High…
## # … with 2,857 more rows, and 7 more variables: partyid <fct>, polviews <fct>,
## #   partners <fct>, pres12 <labelled>, partners_rc <fct>, income16 <fct>,
## #   income_rc <fct>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income")) %>%
  rename(children = childs,
         siblings = sibs) %>%
  relocate(id) %>%
  select(-ballot) %>%
  relocate(where(is.numeric),
           .before = where(is.factor))
```

```
## # A tibble: 2,867 × 18
##       id  year   age children siblings   pres12  region  bigregion degree  race
##    <dbl> <dbl> <dbl>    <dbl> <labelled> <label> <fct>   <fct>     <fct>   <fct>
##  1     1  2016    47        3 2           3      New En… Northeast Bachel… White
##  2     2  2016    61        0 3           1      New En… Northeast High S… White
##  3     3  2016    72        2 3           2      New En… Northeast Bachel… White
##  4     4  2016    43        4 3           2      New En… Northeast High S… White
##  5     5  2016    55        2 2           1      New En… Northeast Gradua… White
##  6     6  2016    53        2 2           1      New En… Northeast Junior… White
##  7     7  2016    50        2 2          NA      New En… Northeast High S… White
##  8     8  2016    23        3 6          NA      Middle… Northeast High S… Other
##  9     9  2016    45        3 5          NA      Middle… Northeast High S… Black
## 10    10  2016    71        4 1           2      Middle… Northeast Junior… White
## # … with 2,857 more rows, and 8 more variables: sex <fct>, padeg <fct>,
## #   partyid <fct>, polviews <fct>, partners <fct>, partners_rc <fct>,
## #   income16 <fct>, income_rc <fct>
```

Shuffle columns around

```
gss_sm %>%
  select(region, bigregion, year,
         id:region,
         starts_with("p"),
         contains("income")) %>%
  rename(children = childs,
         siblings = sibs) %>%
  relocate(id) %>%
  select(-ballot) %>%
  relocate(where(is.numeric),
           .before = where(is.factor)) %>%
  relocate(contains("region"),
           .after = year)
```

```
## # A tibble: 2,867 × 18
##       id  year region   bigregion   age children siblings  pres12  degree  race
##    <dbl> <dbl> <fct>    <fct>     <dbl>    <dbl> <labelle> <label> <fct>   <fct>
##  1     1  2016 New Eng… Northeast    47        3 2          3      Bachel… White
##  2     2  2016 New Eng… Northeast    61        0 3          1      High S… White
##  3     3  2016 New Eng… Northeast    72        2 3          2      Bachel… White
##  4     4  2016 New Eng… Northeast    43        4 3          2      High S… White
##  5     5  2016 New Eng… Northeast    55        2 2          1      Gradua… White
##  6     6  2016 New Eng… Northeast    53        2 2          1      Junior… White
##  7     7  2016 New Eng… Northeast    50        2 2         NA      High S… White
##  8     8  2016 Middle … Northeast    23        3 6         NA      High S… Other
##  9     9  2016 Middle … Northeast    45        3 5         NA      High S… Black
## 10    10  2016 Middle … Northeast    71        4 1          2      Junior… White
## # … with 2,857 more rows, and 8 more variables: sex <fct>, padeg <fct>,
## #   partyid <fct>, polviews <fct>, partners <fct>, partners_rc <fct>,
## #   income16 <fct>, income_rc <fct>
```

Two dplyr gotchas

Comparisons filtering on proportions

Let's say you are working with proportions

```
df
```

```
## # A tibble: 4 × 3
##   id    prop1 prop2
##   <chr> <dbl> <dbl>
## 1 A       0.1   0.2
## 2 B       0.1   0.21
## 3 C       0.11  0.2
## 4 D       0.1   0.1
```

Comparisons filtering on proportions

And you want to focus on cases where `prop1` *plus* `prop2` is greater than 0.3:

Comparisons filtering on proportions

And you want to focus on cases where `prop1` *plus* `prop2` is greater than 0.3:

```
df %>%
  filter(prop1 + prop2 > 0.3)
```

```
## # A tibble: 3 × 3
##   id    prop1 prop2
##   <chr> <dbl> <dbl>
## 1 A       0.1   0.2
## 2 B       0.1   0.21
## 3 C       0.11  0.2
```

`A` shouldn't have been included there.

Comparisons filtering on proportions

And you want to focus on cases where `prop1` *plus* `prop2` is greater than 0.3:

```
df %>%
  filter(prop1 + prop2 > 0.3)
```

```
## # A tibble: 3 × 3
##   id    prop1 prop2
##   <chr> <dbl> <dbl>
## 1 A       0.1   0.2
## 2 B       0.1   0.21
## 3 C       0.11  0.2
```

`A` shouldn't have been included there.

This is not dlpyr's fault. It's our floating point friend again.

Comparisons filtering on proportions

```
df %>%
  filter(prop1 + prop2 == 0.3)
```

```
## # A tibble: 0 × 3
## # … with 3 variables: id <chr>, prop1 <dbl>, prop2 <dbl>
```

A *should* have been included here!

Comparisons filtering on proportions

This won't give the right behavior either:

```
df %>%
  mutate(prop3 = prop1 + prop2) %>%
  filter(prop3 == 0.3)
```

```
## # A tibble: 0 × 4
## # … with 4 variables: id <chr>, prop1 <dbl>, prop2 <dbl>, prop3 <dbl>
```

Comparisons filtering on proportions

So, beware.

```
df %>%
  filter(prop1*100 + prop2*100 == 0.3*100)
```

```
## # A tibble: 1 × 3
##   id    prop1 prop2
##   <chr> <dbl> <dbl>
## 1 A       0.1   0.2
```

Better:

```
df %>%
  filter(near(prop1 + prop2, 0.3))
```

```
## # A tibble: 1 × 3
##   id    prop1 prop2
##   <chr> <dbl> <dbl>
## 1 A       0.1   0.2
```

Zero Counts in dplyr

```
df <- read_csv(here("data", "first_terms.csv"))

df
```

```
## # A tibble: 280 × 4
##      pid start_year party      sex
##    <dbl> <date>     <chr>      <chr>
##  1  3160 2013-01-03 Republican M
##  2  3161 2013-01-03 Democrat   F
##  3  3162 2013-01-03 Democrat   M
##  4  3163 2013-01-03 Republican M
##  5  3164 2013-01-03 Democrat   M
##  6  3165 2013-01-03 Republican M
##  7  3166 2013-01-03 Republican M
##  8  3167 2013-01-03 Democrat   F
##  9  3168 2013-01-03 Republican M
## 10  3169 2013-01-03 Democrat   M
## # … with 270 more rows
```

Zero Counts in dplyr

```
df %>%
    group_by(start_year, party, sex) %>%
    summarize(N = n()) %>%
    mutate(freq = N / sum(N))
```

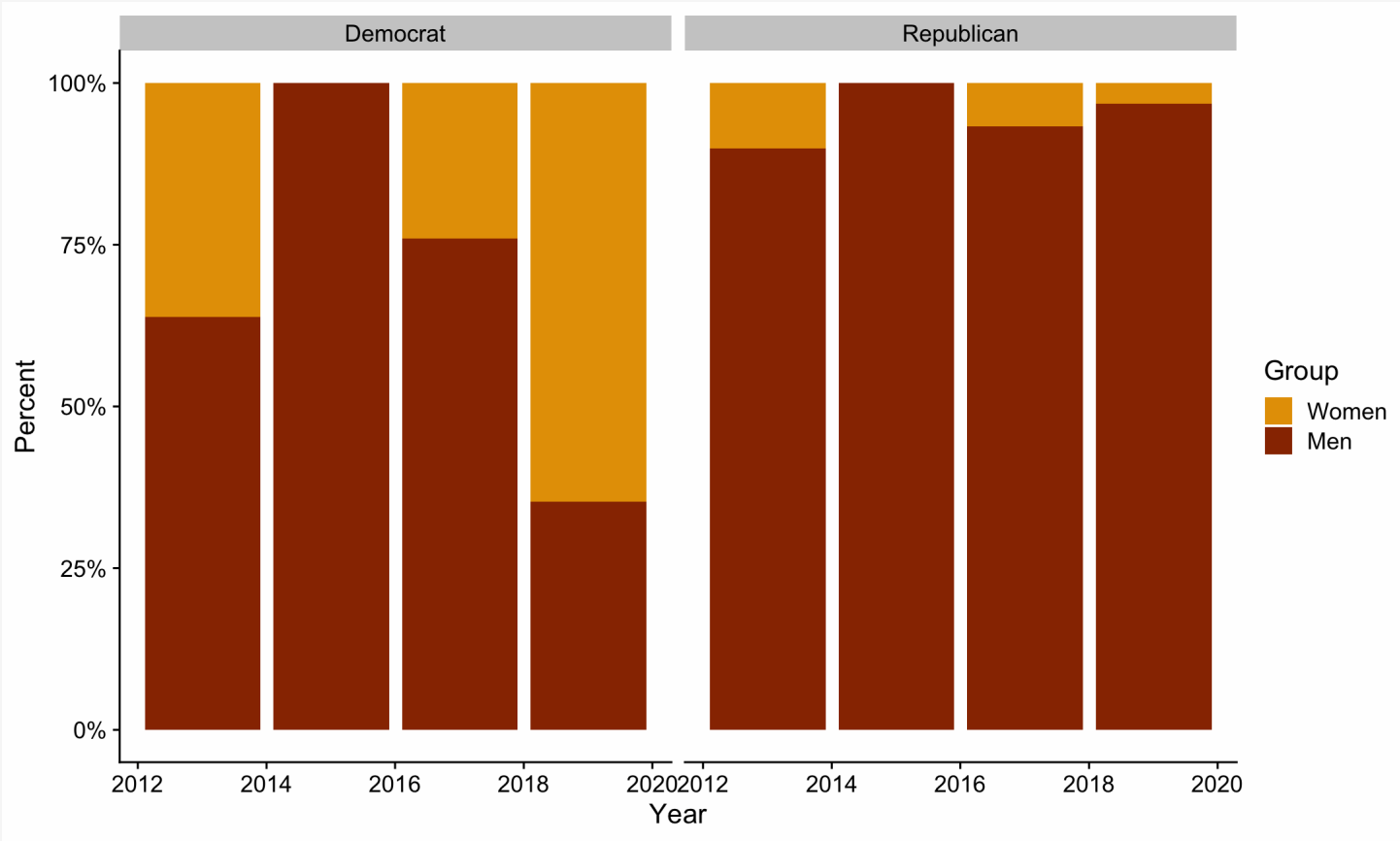
```
## # A tibble: 14 × 5
## # Groups:   start_year, party [8]
##    start_year party      sex       N   freq
##    <date>     <chr>      <chr> <int>  <dbl>
##  1 2013-01-03 Democrat   F        21 0.362
##  2 2013-01-03 Democrat   M        37 0.638
##  3 2013-01-03 Republican F         8 0.101
##  4 2013-01-03 Republican M        71 0.899
##  5 2015-01-03 Democrat   M         1 1
##  6 2015-01-03 Republican M         5 1
##  7 2017-01-03 Democrat   F         6 0.24
##  8 2017-01-03 Democrat   M        19 0.76
##  9 2017-01-03 Republican F         2 0.0667
## 10 2017-01-03 Republican M        28 0.933
## 11 2019-01-03 Democrat   F        33 0.647
## 12 2019-01-03 Democrat   M        18 0.353
## 13 2019-01-03 Republican F         1 0.0323
## 14 2019-01-03 Republican M        30 0.968
```

Zero Counts in dplyr

```
p_col <- df %>%
    group_by(start_year, party, sex) %>%
    summarize(N = n()) %>%
    mutate(freq = N / sum(N)) %>%
    ggplot(aes(x = start_year,
               y = freq,
               fill = sex)) +
    geom_col() +
    scale_y_continuous(labels = scales::percent) +
    scale_fill_manual(values = sex_colors, labels = c("Women", "Men")) +
    labs(x = "Year", y = "Percent", fill = "Group") +
    facet_wrap(~ party)
```

Zero Counts in dplyr

```
p_col
```

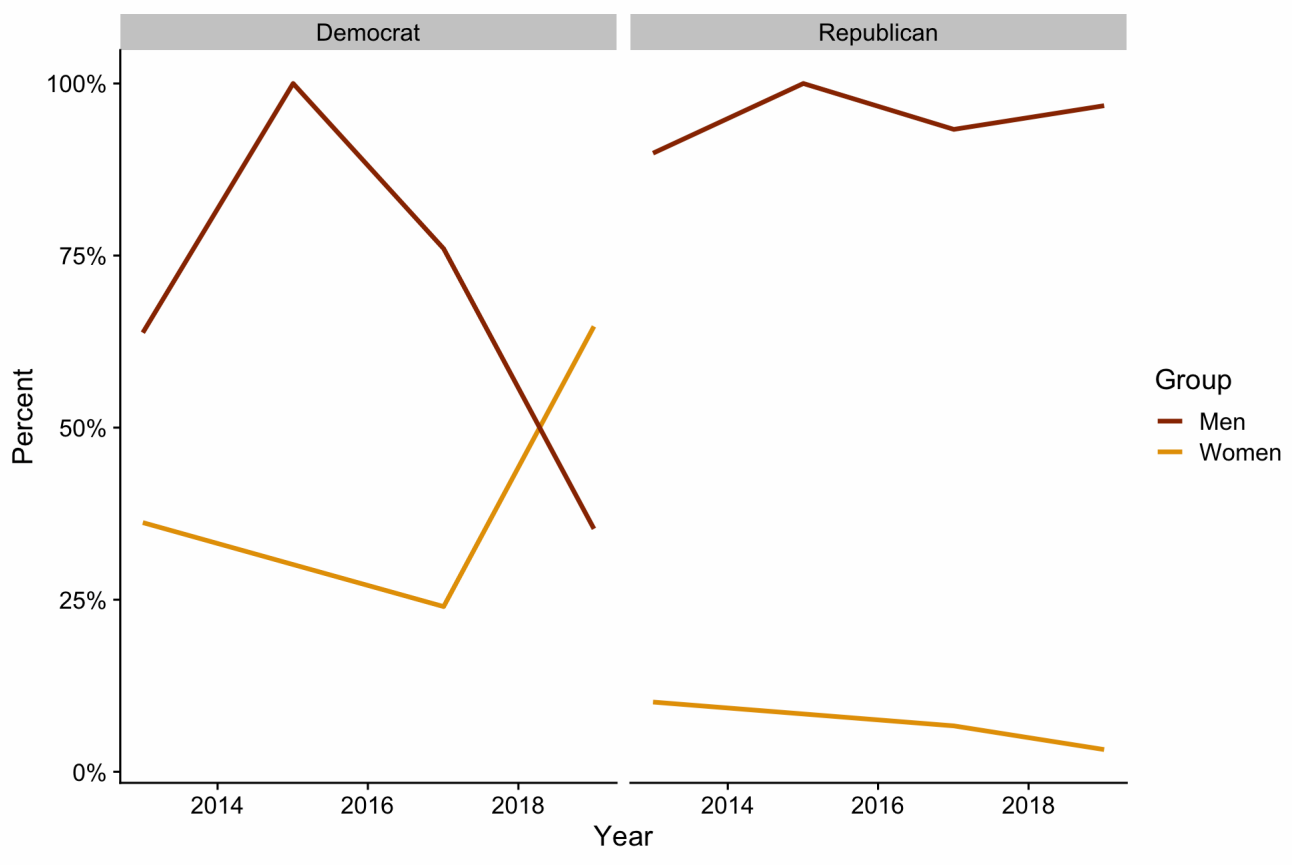


Zero Counts in dplyr

```
p_line <- df %>%
    group_by(start_year, party, sex) %>%
    summarize(N = n()) %>%
    mutate(freq = N / sum(N)) %>%
    ggplot(aes(x = start_year,
               y = freq,
               color = sex)) +
    geom_line(size = 1.1) +
    scale_y_continuous(labels = scales::percent) +
    scale_color_manual(values = sex_colors, labels = c("Women", "Men")) +
    guides(color = guide_legend(reverse = TRUE)) +
    labs(x = "Year", y = "Percent", color = "Group") +
    facet_wrap(~ party)
```

Zero Counts in dplyr

```
p_line
```



Option 1: factors and .drop

Factors are for categorical variables and are stored differently from characters.

This can matter when modeling, and also now.

```
df_f <- df %>%
  mutate(party_f = factor(party))

df_f
```

```
## # A tibble: 280 × 5
##       pid start_year party      sex   party_f
##     <dbl> <date>     <chr>      <chr> <fct>
##  1  3160 2013-01-03 Republican M     Republican
##  2  3161 2013-01-03 Democrat   F     Democrat
##  3  3162 2013-01-03 Democrat   M     Democrat
##  4  3163 2013-01-03 Republican M     Republican
##  5  3164 2013-01-03 Democrat   M     Democrat
##  6  3165 2013-01-03 Republican M     Republican
##  7  3166 2013-01-03 Republican M     Republican
##  8  3167 2013-01-03 Democrat   F     Democrat
##  9  3168 2013-01-03 Republican M     Republican
## 10  3169 2013-01-03 Democrat   M     Democrat
## # … with 270 more rows
```

Option 1: factors and .drop

```
df_f %>%
  group_by(party_f) %>%
  tally()
```

```
## # A tibble: 2 × 2
##   party_f        n
##   <fct>      <int>
## 1 Democrat     135
## 2 Republican   145
```

Factors are integer values with named labels, or *levels*:

```
typeof(df_f$party_f)
```

```
## [1] "integer"
```

```
levels(df_f$party_f)
```

```
## [1] "Democrat"   "Republican"
```

Option 1: factors and .drop

By default, unused levels won't display:

```
df_f <- df %>%
  mutate(party_f = factor(party,
                          levels = c("Democrat",
                                     "Republican",
                                     "Libertarian")))
df_f %>%
  group_by(party_f) %>%
  tally()
```

```
## # A tibble: 2 × 2
##   party_f        n
##   <fct>      <int>
## 1 Democrat     135
## 2 Republican   145
```

```
levels(df_f$party_f)
```

```
## [1] "Democrat"    "Republican"  "Libertarian"
```

Option 1: factors and .drop

By default, unused levels won't display:

```
df %>%
  mutate(across(where(is.character), as_factor)) %>%
  group_by(start_year, party, sex) %>%
  summarize(N = n()) %>%
  mutate(freq = N / sum(N))
```

```
## # A tibble: 14 × 5
## # Groups:   start_year, party [8]
##    start_year party      sex       N   freq
##    <date>     <fct>      <fct> <int>  <dbl>
##  1 2013-01-03 Republican M        71 0.899
##  2 2013-01-03 Republican F         8 0.101
##  3 2013-01-03 Democrat   M        37 0.638
##  4 2013-01-03 Democrat   F        21 0.362
##  5 2015-01-03 Republican M         5 1
##  6 2015-01-03 Democrat   M         1 1
##  7 2017-01-03 Republican M        28 0.933
##  8 2017-01-03 Republican F         2 0.0667
##  9 2017-01-03 Democrat   M        19 0.76
## 10 2017-01-03 Democrat   F         6 0.24
## 11 2019-01-03 Republican M        30 0.968
## 12 2019-01-03 Republican F         1 0.0323
```

Option 1: factors and .drop

You can make `dplyr` keep empty factor levels though:

```
df %>%
  mutate(across(where(is.character), as_factor)) %>%
  group_by(start_year, party, sex, .drop = FALSE) %>%
  summarize(N = n()) %>%
  mutate(freq = N / sum(N))
```

```
## # A tibble: 16 × 5
## # Groups:   start_year, party [8]
##    start_year party      sex       N   freq
##    <date>     <fct>      <fct> <int>  <dbl>
##  1 2013-01-03 Republican M        71 0.899
##  2 2013-01-03 Republican F         8 0.101
##  3 2013-01-03 Democrat   M        37 0.638
##  4 2013-01-03 Democrat   F        21 0.362
##  5 2015-01-03 Republican M         5 1
##  6 2015-01-03 Republican F         0 0
##  7 2015-01-03 Democrat   M         1 1
##  8 2015-01-03 Democrat   F         0 0
##  9 2017-01-03 Republican M        28 0.933
## 10 2017-01-03 Republican F         2 0.0667
## 11 2017-01-03 Democrat   M        19 0.76
## 12 2017-01-03 Democrat   F         6 0.24
```

Option 2: ungroup() and complete()

Maybe you don't want to deal with factors.

```
df_c <- df %>%
    group_by(start_year, party, sex) %>%
    summarize(N = n()) %>%
    mutate(freq = N / sum(N)) %>%
    ungroup() %>%
    complete(start_year, party, sex,
             fill = list(N = 0, freq = 0))
```

Option 2: ungroup() and complete()

```
df_c
```

```
## # A tibble: 16 × 5
##    start_year party      sex       N   freq
##    <date>     <chr>      <chr> <dbl>  <dbl>
##  1 2013-01-03 Democrat   F        21 0.362
##  2 2013-01-03 Democrat   M        37 0.638
##  3 2013-01-03 Republican F         8 0.101
##  4 2013-01-03 Republican M        71 0.899
##  5 2015-01-03 Democrat   F         0 0
##  6 2015-01-03 Democrat   M         1 1
##  7 2015-01-03 Republican F         0 0
##  8 2015-01-03 Republican M         5 1
##  9 2017-01-03 Democrat   F         6 0.24
## 10 2017-01-03 Democrat   M        19 0.76
## 11 2017-01-03 Republican F         2 0.0667
## 12 2017-01-03 Republican M        28 0.933
## 13 2019-01-03 Democrat   F        33 0.647
## 14 2019-01-03 Democrat   M        18 0.353
## 15 2019-01-03 Republican F         1 0.0323
## 16 2019-01-03 Republican M        30 0.968
```

Option 2: ungroup() and complete()

```
p_out <- df_c %>%
  ggplot(aes(x = start_year,
             y = freq,
             color = sex)) +
    geom_line(size = 1.1) +
    scale_y_continuous(labels = scales::percent) +
    scale_color_manual(values = sex_colors, labels = c("Women", "Men")) +
    guides(color = guide_legend(reverse = TRUE)) +
    labs(x = "Year", y = "Percent", color = "Group") +
    facet_wrap(~ party)
```

Option 2: ungroup() and complete()

```
p_out
```