Manipulating Tables with dplyr

Session 3

Kieran Healy

Statistical Horizons, April 2021

Time to play with some data

Time to play with some data

woohoo!

Load our libraries

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

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

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

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

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

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

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──
```

```
## ✓ ggplot2 3.3.3     ✓ purrr   0.3.4
## ✓ tibble  3.1.0     ✓ dplyr   1.0.5
## ✓ tidyr   1.1.3     ✓ stringr 1.4.0
## ✓ readr   1.4.0     ✓ forcats 0.5.1
```

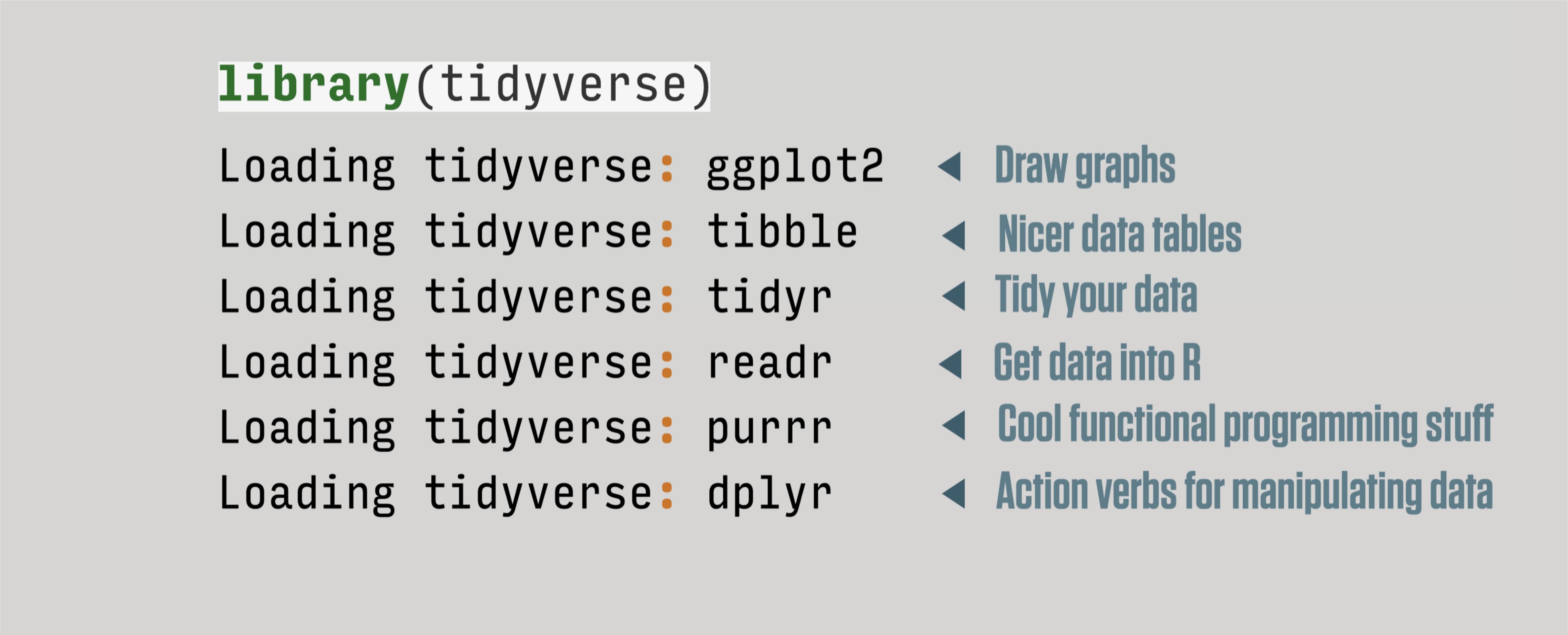
```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter()  masks stats::filter()
## x purrr::is_null() masks testthat::is_null()
## x dplyr::lag()     masks stats::lag()
## x dplyr::matches() masks tidyr::matches(), testthat::matches()
```

Tidyverse components, again

```r
library(tidyverse)
```

```
Loading tidyverse: ggplot2
Loading tidyverse: tibble
Loading tidyverse: tidyr
Loading tidyverse: readr
Loading tidyverse: purrr
Loading tidyverse: dplyr
```

Tidyverse components, again



```r
library(tidyverse)
```

```
Loading tidyverse: ggplot2
Loading tidyverse: tibble
Loading tidyverse: tidyr
Loading tidyverse: readr
Loading tidyverse: purrr
Loading tidyverse: dplyr
```

◀ Draw graphs

◀ Nicer data tables

◀ Tidy your data

◀ Get data into R

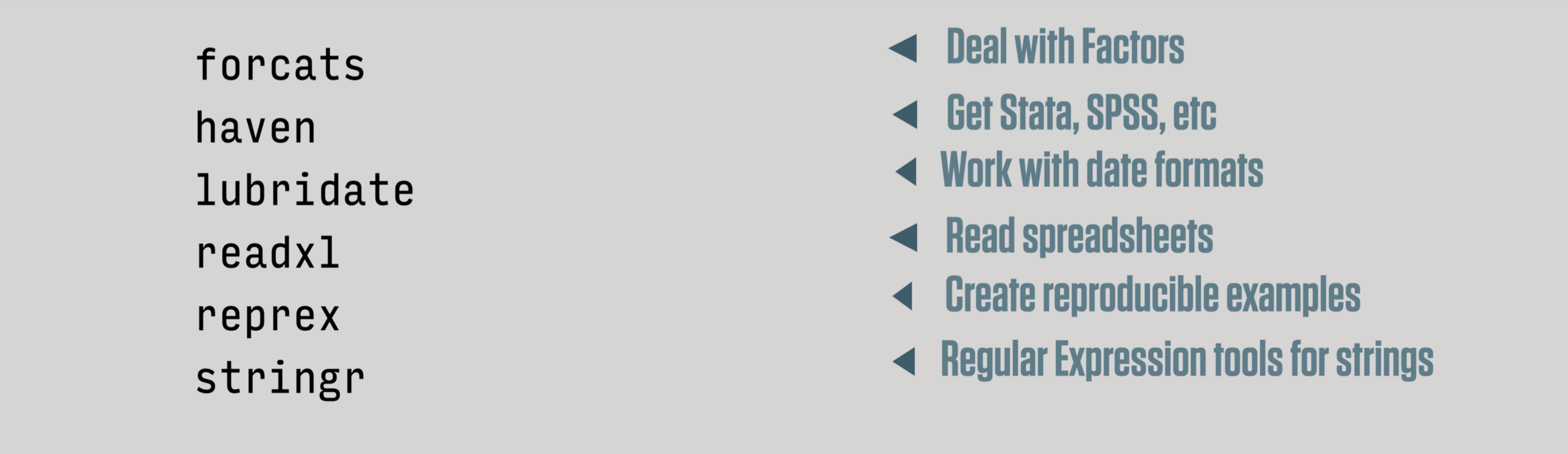
◀ Cool functional programming stuff

◀ Action verbs for manipulating data

Other tidyverse components

```
forcats
haven
lubridate
readxl
reprex
stringr
```

Other tidyverse components



`forcats`	◀ Deal with Factors
`haven`	◀ Get Stata, SPSS, etc
`lubridate`	◀ Work with date formats
`readxl`	◀ Read spreadsheets
`reprex`	◀ Create reproducible examples
`stringr`	◀ Regular Expression tools for strings

Not all of these are attached when we do `library(tidyverse)`

dplyr lets you work with tibbles

Remember, tibbles are tables of data where the columns can be of different types, such as numeric, logical, character, factor, etc.

dplyr lets you work with tibbles

Remember, tibbles are tables of data where the columns can be of different types, such as numeric, logical, character, factor, etc.

We'll use dplyr to *transform* and *summarize* our data.

dplyr lets you work with tibbles

Remember, tibbles are tables of data where the columns can be of different types, such as numeric, logical, character, factor, etc.

We'll use dplyr to *transform* and *summarize* our data.

We'll use the pipe operator, `%>%`, to chain together sequences of actions on our tables.

dplyr lets you work with tibbles

Remember, tibbles are tables of data where the columns can be of different types, such as numeric, logical, character, factor, etc.

We'll use dplyr to *transform* and *summarize* our data.

We'll use the pipe operator, **%>%**, to chain together sequences of actions on our tables.

`dplyr` draws on the logic and language of database queries, where the focus is on manipulating tables

Some actions to take on a single table

Some actions to take on a single table

Group the data at the level we want, such as "*Religion by Region*" or "*Children by School by District*", so as to present data at that level.

Some actions to take on a single table

Group the data at the level we want, such as “*Religion by Region*” or “*Children by School by District*”, so as to present data at that level.

Subset either the rows or columns of or table.

Some actions to take on a single table

Group the data at the level we want, such as “*Religion by Region*” or “*Children by School by District*”, so as to present data at that level.

Subset either the rows or columns of or table.

Mutate the data. That is, change something at the *current* level of grouping. Mutating adds new columns to the table, or changes the content of an existing column. This won't change the number of rows.

Some actions to take on a single table

Group the data at the level we want, such as "*Religion by Region*" or "*Children by School by District*", so as to present data at that level.

Subset either the rows or columns of or table.

Mutate the data. That is, change something at the *current* level of grouping. Mutating adds new columns to the table, or changes the content of an existing column. This won't change the number of rows.

Summarize or aggregate the data. That is, make something new at a *higher* level of grouping. E.g., calculate means or counts by some grouping variable. This will generally result in a smaller, *summary* table.

Each action has a corresponding function

Each action has a corresponding function

Group using `group_by()`.

Each action has a corresponding function

Group using `group_by()`.

Subset has one action for rows and one for columns. We `filter()` rows and `select()` columns.

Each action has a corresponding function

Group using `group_by()`.

Subset has one action for rows and one for columns. We `filter()` rows and `select()` columns.

Mutate tables (i.e. add new columns, or re-make existing ones) using `mutate()`.

Each action has a corresponding function

Group using `group_by()`.

Subset has one action for rows and one for columns. We `filter()` rows and `select()` columns.

Mutate tables (i.e. add new columns, or re-make existing ones) using `mutate()`.

Summarize tables (i.e. perform aggregating calculations) using `summarize()`.

General Social Survey data: gss_sm

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

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

General Social Survey data: `gss_sm`

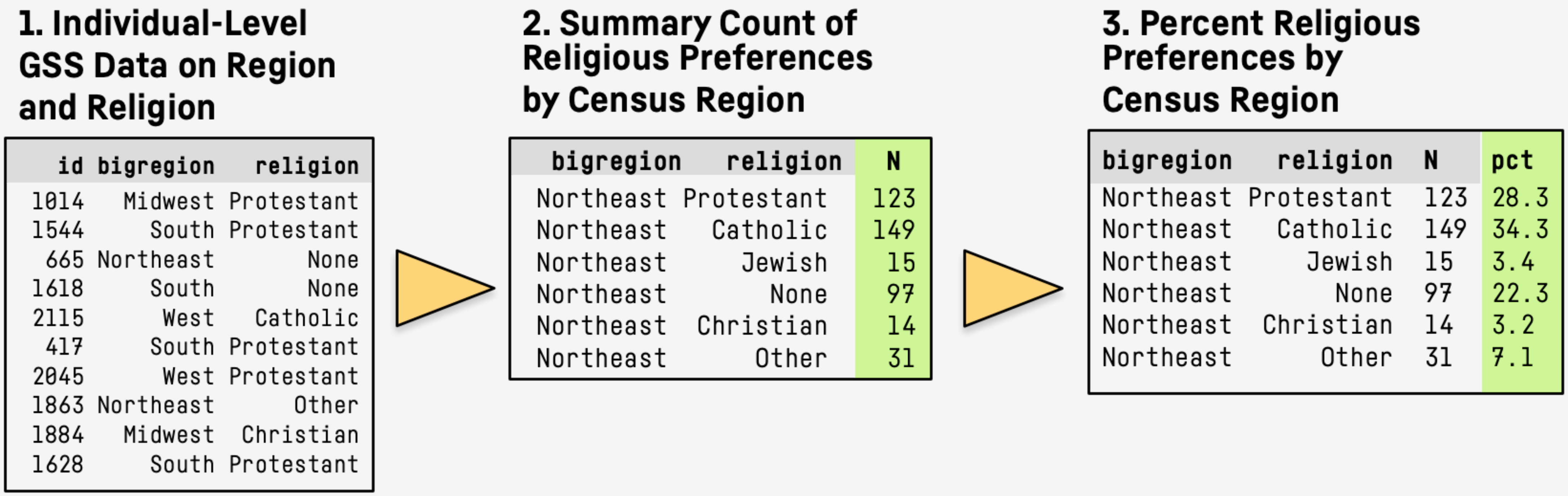
```r
## library(socviz) # if not loaded
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Notice how the tibble already tells us a lot.

Summarizing a Table

Here's what we're going to do:



1. Individual-Level GSS Data on Region and Religion

id	bigregion	religion
1014	Midwest	Protestant
1544	South	Protestant
665	Northeast	None
1618	South	None
2115	West	Catholic
417	South	Protestant
2045	West	Protestant
1863	Northeast	Other
1884	Midwest	Christian
1628	South	Protestant

2. Summary Count of Religious Preferences by Census Region

bigregion	religion	N
Northeast	Protestant	123
Northeast	Catholic	149
Northeast	Jewish	15
Northeast	None	97
Northeast	Christian	14
Northeast	Other	31

3. Percent Religious Preferences by Census Region

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

Summarizing a table

```
## Just take a look at the columns we will work on
gss_sm %>%
  select(id, bigregion, religion)
```

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

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

Group by *one* column or variable

```r
gss_sm %>%
  group_by(bigregion)
```

```
## # A tibble: 2,867 x 32
## # Groups:   bigregion [4]
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Grouping just changes the logical structure of the tibble.

Group and summarize by *one* column

```
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *one* column

```r
gss_sm %>%
  group_by(bigregion)
```

```
## # A tibble: 2,867 x 32
## # Groups:   bigregion [4]
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##   1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##   2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##   3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##   4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##   5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##   6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##   7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##   8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##   9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *one* column

```r
gss_sm %>%
  group_by(bigregion) %>%
  summarize(total = n())
```

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

Group and summarize by *one* column

```r
gss_sm %>%
  group_by(bigregion) %>%
  summarize(total = n())
```

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

The function `n()` counts up the rows within each group.

Group and summarize by *one* column

```r
gss_sm %>%
  group_by(bigregion) %>%
  summarize(total = n())
```

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

The function `n()` counts up the rows within each group.

All the other columns are dropped in the summary operation

Group and summarize by *one* column

```
gss_sm %>%
  group_by(bigregion) %>%
  summarize(total = n())
```

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

The function `n()` counts up the rows within each group.

All the other columns are dropped in the summary operation

Your original `gss_sm` table is untouched

Group and summarize by *two* columns

```
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *two* columns

```
gss_sm %>%
  group_by(bigregion, religion)
```

```
## # A tibble: 2,867 x 32
## # Groups:   bigregion, religion [24]
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Group and summarize by *two* columns

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

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion   total
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

Group and summarize by *two* columns

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

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion   total
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

The function `n()` counts up the rows within the *innermost* (i.e. the rightmost) group.

Calculate frequencies

```
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Calculate frequencies

```r
gss_sm %>%
  group_by(bigregion, religion)
```

```
## # A tibble: 2,867 x 32
## # Groups:   bigregion, religion [24]
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>   
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Calculate frequencies

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

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion   total
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

Calculate frequencies

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23  
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

Calculate frequencies

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23  
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

The function `n()` counts up the rows

Calculate frequencies

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23  
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

The function `n()` counts up the rows

Which rows? The ones fed down the pipeline

Calculate frequencies

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

The function `n()` counts up the rows

Which rows? The ones fed down the pipeline

The *innermost* (i.e. the rightmost) group.

Pipelines carry some assumptions forward

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

Groups are carried forward till summarized or explicitly ungrouped

Pipelines carry some assumptions forward

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23  
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

Groups are carried forward till summarized or explicitly ungrouped

Summary calculations are done on the innermost group, which then "disappears". (Notice how it's no longer a group in the output.)

Pipelines carry some assumptions forward

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

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

Convenience functions

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

```
## # A tibble: 24 x 5
## # Groups:   bigregion [4]
##    bigregion religion   total    freq   pct
##    <fct>     <fct>      <int>   <dbl> <dbl>
##  1 Northeast Protestant   158 0.324    32.4
##  2 Northeast Catholic     162 0.332    33.2
##  3 Northeast Jewish        27 0.0553    5.5
##  4 Northeast None         112 0.230    23  
##  5 Northeast Other         28 0.0574    5.7
##  6 Northeast <NA>           1 0.00205   0.2
##  7 Midwest   Protestant   325 0.468    46.8
##  8 Midwest   Catholic     172 0.247    24.7
##  9 Midwest   Jewish         3 0.00432   0.4
## 10 Midwest   None         157 0.226    22.6
## # … with 14 more rows
```

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

Three options for counting up rows

Do it yourself

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  summarize(n = n())
```

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion       n
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

Result is a grouped tibble

Three options for counting up rows

Do it yourself

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  summarize(n = n())
```

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion        n
##    <fct>     <fct>       <int>
##  1 Northeast Protestant    158
##  2 Northeast Catholic      162
##  3 Northeast Jewish         27
##  4 Northeast None          112
##  5 Northeast Other          28
##  6 Northeast <NA>            1
##  7 Midwest   Protestant    325
##  8 Midwest   Catholic      172
##  9 Midwest   Jewish          3
## 10 Midwest   None          157
## # … with 14 more rows
```

Result is a grouped tibble

use `tally()`

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  tally()
```

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion        n
##    <fct>     <fct>       <int>
##  1 Northeast Protestant    158
##  2 Northeast Catholic      162
##  3 Northeast Jewish         27
##  4 Northeast None          112
##  5 Northeast Other          28
##  6 Northeast <NA>            1
##  7 Midwest   Protestant    325
##  8 Midwest   Catholic      172
##  9 Midwest   Jewish          3
## 10 Midwest   None          157
## # … with 14 more rows
```

Group it yourself; output is grouped

Three options for counting up rows

Do it yourself

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  summarize(n = n())
```

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion       n
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

Result is a grouped tibble

use `tally()`

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  tally()
```

```
## # A tibble: 24 x 3
## # Groups:   bigregion [4]
##    bigregion religion       n
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

Group it yourself; output is grouped

use `count()`

```
gss_sm %>%
  count(bigregion, religion)
```

```
## # A tibble: 24 x 3
##    bigregion religion       n
##    <fct>     <fct>      <int>
##  1 Northeast Protestant   158
##  2 Northeast Catholic     162
##  3 Northeast Jewish        27
##  4 Northeast None         112
##  5 Northeast Other         28
##  6 Northeast <NA>           1
##  7 Midwest   Protestant   325
##  8 Midwest   Catholic     172
##  9 Midwest   Jewish         3
## 10 Midwest   None         157
## # … with 14 more rows
```

One step; result is not grouped

Pass your pipeline on to ... a table

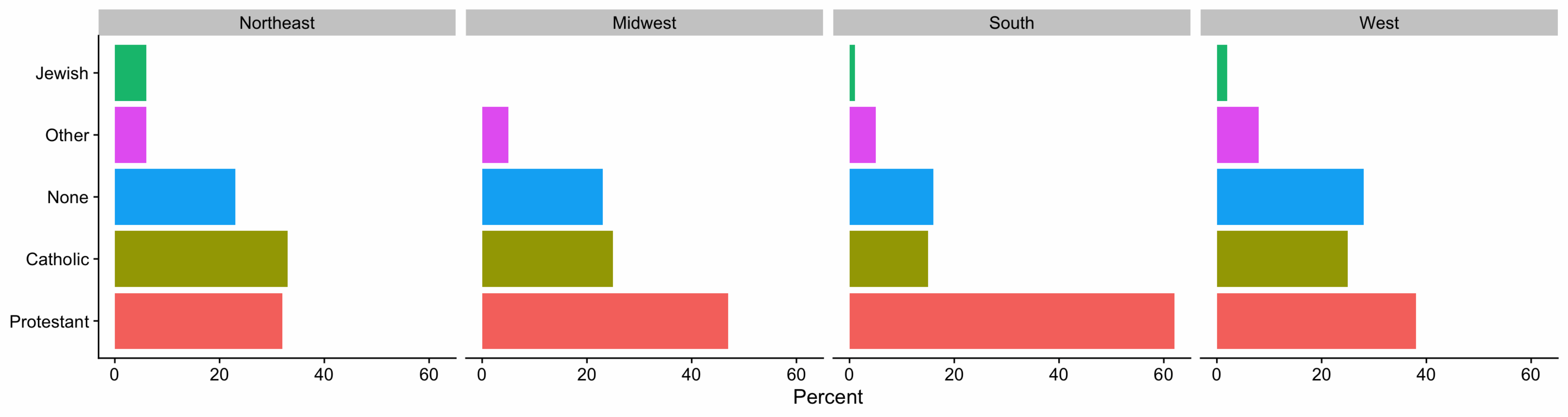
```
gss_sm %>%
  count(bigregion, religion) %>%
  pivot_wider(names_from = bigregion, values_from = n) %>%
  kable()
```

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

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

Pass your pipeline on to ... a graph

```
gss_sm %>%
  group_by(bigregion, religion) %>%
  tally() %>%
  mutate(pct = round((n/sum(n))*100), 1) %>%
  drop_na() %>%
  ggplot(mapping = aes(x = pct, y = reorder(religion, -pct), fill = religion)) +
  geom_col() +
    labs(x = "Percent", y = NULL) +
    guides(fill = FALSE) +
    facet_wrap(~ bigregion, nrow = 1)
```



Pass your pipeline on to ... an **object**

You can do it like this ...

```r
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

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

Pass your pipeline on to ... an **object**

You can do it like this ...

```
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

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

Or like this!

```
gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1)) ->
rel_by_region

rel_by_region
```

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

Right assignmment is a thing, just like Left

Left assignment is standard

```
gss_tab <- gss_sm %>%
  count(bigregion, religion)
```

This may feel awkward with a pipe: "`gss_tab` *gets* the output of the following pipeline."

Right assignmment is a thing, just like Left

Left assignment is standard

```r
gss_tab <- gss_sm %>%
  count(bigregion, religion)
```

This may feel awkward with a pipe: "`gss_tab` *gets* the output of the following pipeline."

Right assignment also works!

```r
gss_sm %>%
  count(bigregion, religion) -> gss_tab
```

Without any authority, I assert that right-assignment should be read as, e.g., "This pipeline *begets* `gss_tab`"

Pipelined tables can be quickly checked

```
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

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

Hm, did I sum over right group?

Pipelined tables can be quickly checked

```
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))

rel_by_region
```

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

Hm, did I sum over right group?

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

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

No! What has gone wrong here?

Pipelined tables can be quickly checked

```
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))
```

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

```
rel_by_region %>%
  summarize(total = sum(pct))
```

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

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

Pipelined tables can be quickly checked

```
rel_by_region <- gss_sm %>%
  count(bigregion, religion) %>%
  mutate(pct = round((n/sum(n))*100, 1))
```

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

```
rel_by_region %>%
  summarize(total = sum(pct))
```

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

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

```
rel_by_region <- gss_sm %>%
  group_by(bigregion, religion) %>%
  tally() %>%
  mutate(pct = round((n/sum(n))*100, 1))
```

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

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

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

Two lessons

Check your tables!

> Pipelines feed their content forward, so you need to make sure your results are not incorrect.

Two lessons

Check your tables!

Pipelines feed their content forward, so you need to make sure your results are not incorrect.

Often, complex tables and graphs can be disturbingly plausible even when wrong.

Two lessons

Check your tables!

Pipelines feed their content forward, so you need to make sure your results are not incorrect.

Often, complex tables and graphs can be disturbingly plausible even when wrong.

So, figure out what the result should be and test it!

Two lessons

Check your tables!

Pipelines feed their content forward, so you need to make sure your results are not incorrect.

Often, complex tables and graphs can be disturbingly plausible even when wrong.

So, figure out what the result should be and test it!

Starting with simple or toy cases can help with this process.

Two lessons

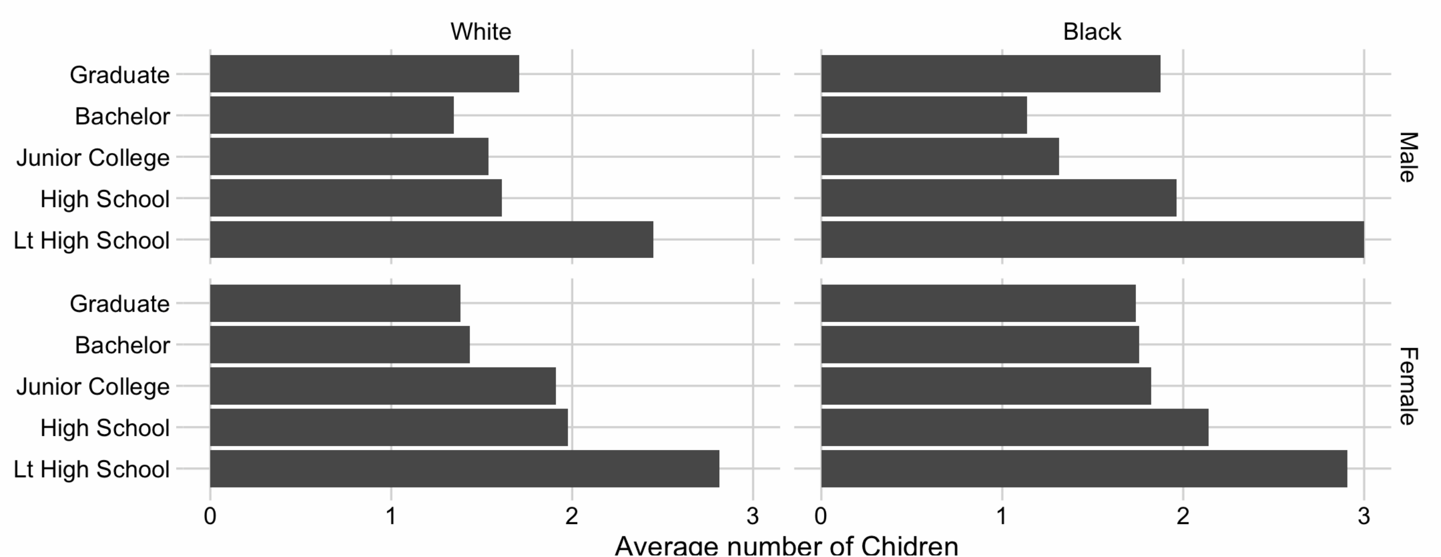
Inspect your pipes!

Understand pipelines by running them forward or peeling them back a step at a time.

This is a *very* effective way to understand your own and other people's code.

Following a pipeline

```
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE)) %>%
  mutate(pct = n/sum(n)*100) %>%
  filter(race !="Other") %>%
  drop_na() %>%
  ggplot(mapping = aes(x = mean_kids, y = degree)) + # I'm sorry I can't talk more about the graphs
  geom_col() + facet_grid(sex ~ race) +
  labs(x = "Average number of Chidren", y = NULL)
```



Following a pipeline

```
gss_sm
```

```
## # A tibble: 2,867 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Following a pipeline

```
gss_sm %>%
  group_by(race, sex, degree)
```

```
## # A tibble: 2,867 x 32
## # Groups:   race, sex, degree [34]
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>
```

Following a pipeline

```
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE))
```

```
## # A tibble: 34 x 6
## # Groups:   race, sex [6]
##    race  sex    degree             n mean_age mean_kids
##    <fct> <fct>  <fct>          <int>    <dbl>     <dbl>
##  1 White Male   Lt High School    96     52.9      2.45
##  2 White Male   High School      470     48.8      1.61
##  3 White Male   Junior College    65     47.1      1.54
##  4 White Male   Bachelor         208     48.6      1.35
##  5 White Male   Graduate         112     56.0      1.71
##  6 White Female Lt High School   101     55.4      2.81
##  7 White Female High School      587     51.9      1.98
##  8 White Female Junior College   101     48.2      1.91
##  9 White Female Bachelor         218     49.2      1.44
## 10 White Female Graduate         138     53.6      1.38
## # … with 24 more rows
```

Following a pipeline

```r
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE)) %>%
  mutate(pct = n/sum(n)*100)
```

```
## # A tibble: 34 x 7
## # Groups:   race, sex [6]
##    race  sex    degree              n mean_age mean_kids   pct
##    <fct> <fct>  <fct>           <int>    <dbl>     <dbl> <dbl>
##  1 White Male   Lt High School     96     52.9      2.45 10.1
##  2 White Male   High School       470     48.8      1.61 49.4
##  3 White Male   Junior College     65     47.1      1.54  6.83
##  4 White Male   Bachelor          208     48.6      1.35 21.9
##  5 White Male   Graduate          112     56.0      1.71 11.8
##  6 White Female Lt High School    101     55.4      2.81  8.79
##  7 White Female High School       587     51.9      1.98 51.1
##  8 White Female Junior College    101     48.2      1.91  8.79
##  9 White Female Bachelor          218     49.2      1.44 19.0
## 10 White Female Graduate          138     53.6      1.38 12.0
## # … with 24 more rows
```

Following a pipeline

```
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE)) %>%
  mutate(pct = n/sum(n)*100) %>%
  filter(race !="Other")
```

```
## # A tibble: 23 x 7
## # Groups:   race, sex [4]
##    race  sex    degree             n mean_age mean_kids   pct
##    <fct> <fct>  <fct>          <int>    <dbl>     <dbl> <dbl>
##  1 White Male   Lt High School    96     52.9      2.45 10.1 
##  2 White Male   High School      470     48.8      1.61 49.4 
##  3 White Male   Junior College    65     47.1      1.54  6.83
##  4 White Male   Bachelor         208     48.6      1.35 21.9 
##  5 White Male   Graduate         112     56.0      1.71 11.8 
##  6 White Female Lt High School   101     55.4      2.81  8.79
##  7 White Female High School      587     51.9      1.98 51.1 
##  8 White Female Junior College   101     48.2      1.91  8.79
##  9 White Female Bachelor         218     49.2      1.44 19.0 
## 10 White Female Graduate         138     53.6      1.38 12.0 
## # … with 13 more rows
```

Following a pipeline

```r
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE)) %>%
  mutate(pct = n/sum(n)*100) %>%
  filter(race !="Other") %>%
  drop_na()
```

```
## # A tibble: 20 x 7
## # Groups:   race, sex [4]
##    race  sex    degree             n mean_age mean_kids   pct
##    <fct> <fct>  <fct>          <int>    <dbl>     <dbl> <dbl>
##  1 White Male   Lt High School    96     52.9      2.45 10.1
##  2 White Male   High School      470     48.8      1.61 49.4
##  3 White Male   Junior College    65     47.1      1.54  6.83
##  4 White Male   Bachelor         208     48.6      1.35 21.9
##  5 White Male   Graduate         112     56.0      1.71 11.8
##  6 White Female Lt High School   101     55.4      2.81  8.79
##  7 White Female High School      587     51.9      1.98 51.1
##  8 White Female Junior College   101     48.2      1.91  8.79
##  9 White Female Bachelor         218     49.2      1.44 19.0
## 10 White Female Graduate         138     53.6      1.38 12.0
## 11 Black Male   Lt High School    17     56.1      3     8.21
## 12 Black Male   High School      142     43.6      1.96 68.6
## 13 Black Male   Junior College    16     47.1      1.31  7.73
## 14 Black Male   Bachelor          22     41.6      1.14 10.6
## 15 Black Male   Graduate           8     53.1      1.88  3.86
## 16 Black Female Lt High School    43     51.0      2.91 15.2
## 17 Black Female High School      150     43.1      2.14 53.0
## 18 Black Female Junior College    17     45.8      1.82  6.01
## 19 Black Female Bachelor          49     47.0      1.76 17.3
## 20 Black Female Graduate          23     51.2      1.74  8.13
```

Following a pipeline

```
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE)) %>%
  mutate(pct = n/sum(n)*100) %>%
  filter(race !="Other") %>%
  drop_na() %>%
  summarize(grp_totpct = sum(pct))
```

```
## # A tibble: 4 x 3
## # Groups:   race [2]
##   race  sex    grp_totpct
##   <fct> <fct>       <dbl>
## 1 White Male        100
## 2 White Female       99.7
## 3 Black Male         99.0
## 4 Black Female       99.6
```

Conditionals in `select()` and `filter()`

Some new data, this time on national rates of cadaveric organ donation:

```r
# library(socviz)
organdata
```

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

Conditionals in `select()` and `filter()`

```r
organdata %>%
  filter(consent_law == "Informed" & donors > 15)
```

```
## # A tibble: 30 x 21
##    country year       donors   pop pop_dens   gdp gdp_lag health health_lag
##    <chr>   <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Canada  2000-01-01   15.3 30770    0.309 28472   26658   2541       2400
##  2 Denmark 1992-01-01   16.1  5171   12.0   19644   19126   1660       1603
##  3 Ireland 1991-01-01   19     3534    5.03 13495   12917    884        791
##  4 Ireland 1992-01-01   19.5  3558    5.06 14241   13495   1005        884
##  5 Ireland 1993-01-01   17.1  3576    5.09 14927   14241   1041       1005
##  6 Ireland 1994-01-01   20.3  3590    5.11 15990   14927   1119       1041
##  7 Ireland 1995-01-01   24.6  3609    5.14 17789   15990   1208       1119
##  8 Ireland 1996-01-01   16.8  3636    5.17 19245   17789   1269       1208
##  9 Ireland 1997-01-01   20.9  3673    5.23 22017   19245   1417       1269
## 10 Ireland 1998-01-01   23.8  3715    5.29 23995   22017   1487       1417
## # … with 20 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

Conditionals in `select()` and `filter()`

```r
organdata %>%
  select(country, year, where(is.integer))
```

```
## # A tibble: 238 x 8
##    country   year         pop   gdp gdp_lag cerebvas assault external
##    <chr>     <date>     <int> <int>   <int>    <int>   <int>    <int>
##  1 Australia NA         17065 16774   16591      682      21      444
##  2 Australia 1991-01-01 17284 17171   16774      647      19      425
##  3 Australia 1992-01-01 17495 17914   17171      630      17      406
##  4 Australia 1993-01-01 17667 18883   17914      611      18      376
##  5 Australia 1994-01-01 17855 19849   18883      631      17      387
##  6 Australia 1995-01-01 18072 21079   19849      592      16      371
##  7 Australia 1996-01-01 18311 21923   21079      576      17      395
##  8 Australia 1997-01-01 18518 22961   21923      525      17      385
##  9 Australia 1998-01-01 18711 24148   22961      516      16      410
## 10 Australia 1999-01-01 18926 25445   24148      493      15      409
## # … with 228 more rows
```

Use `where()` to test columns.

Conditionals in `select()` and `filter()`

When telling `where()` use `is.integer()` to test each column, we don't put parentheses at the end of its name. If we did, R would try to evaluate `is.integer()` right then, and fail:

```
> organdata %>%
+   select(country, year, where(is.integer()))
Error: 0 arguments passed to 'is.integer' which requires 1
Run `rlang::last_error()` to see where the error occurred.
```

This is true in similar situations elsewhere as well.

Conditionals in `select()` and `filter()`

```
organdata %>%
  select(country, year, where(is.character))
```

```
## # A tibble: 238 x 8
##    country  year       world opt   consent_law consent_practice consistent ccode
##    <chr>    <date>     <chr> <chr> <chr>       <chr>            <chr>      <chr>
##  1 Austral… NA         Libe… In    Informed    Informed         Yes        Oz
##  2 Austral… 1991-01-01 Libe… In    Informed    Informed         Yes        Oz
##  3 Austral… 1992-01-01 Libe… In    Informed    Informed         Yes        Oz
##  4 Austral… 1993-01-01 Libe… In    Informed    Informed         Yes        Oz
##  5 Austral… 1994-01-01 Libe… In    Informed    Informed         Yes        Oz
##  6 Austral… 1995-01-01 Libe… In    Informed    Informed         Yes        Oz
##  7 Austral… 1996-01-01 Libe… In    Informed    Informed         Yes        Oz
##  8 Austral… 1997-01-01 Libe… In    Informed    Informed         Yes        Oz
##  9 Austral… 1998-01-01 Libe… In    Informed    Informed         Yes        Oz
## 10 Austral… 1999-01-01 Libe… In    Informed    Informed         Yes        Oz
## # … with 228 more rows
```

We have functions like e.g. `is.character()`, `is.numeric()`, `is.logical()`, `is.factor()`, etc. All return either `TRUE` or `FALSE`.

Conditionals in `select()` and `filter()`

Sometimes we don't pass a function, but do want to use the result of one:

```
organdata %>%
  select(country, year, starts_with("gdp"))
```

```
## # A tibble: 238 x 4
##    country   year         gdp gdp_lag
##    <chr>     <date>     <int>   <int>
##  1 Australia NA         16774   16591
##  2 Australia 1991-01-01 17171   16774
##  3 Australia 1992-01-01 17914   17171
##  4 Australia 1993-01-01 18883   17914
##  5 Australia 1994-01-01 19849   18883
##  6 Australia 1995-01-01 21079   19849
##  7 Australia 1996-01-01 21923   21079
##  8 Australia 1997-01-01 22961   21923
##  9 Australia 1998-01-01 24148   22961
## 10 Australia 1999-01-01 25445   24148
## # … with 228 more rows
```

We have `starts_with()`, `ends_with()`, `contains()`, `matches()`, and `num_range()`. Collectively these are "tidy selectors".

Conditionals in `select()` and `filter()`

```r
organdata %>%
  filter(country == "Australia" | country == "Canada")
```

```
## # A tibble: 28 x 21
##    country   year       donors    pop pop_dens   gdp gdp_lag health health_lag
##    <chr>     <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Australia NA          NA    17065    0.220 16774   16591   1300       1224
##  2 Australia 1991-01-01  12.1  17284    0.223 17171   16774   1379       1300
##  3 Australia 1992-01-01  12.4  17495    0.226 17914   17171   1455       1379
##  4 Australia 1993-01-01  12.5  17667    0.228 18883   17914   1540       1455
##  5 Australia 1994-01-01  10.2  17855    0.231 19849   18883   1626       1540
##  6 Australia 1995-01-01  10.2  18072    0.233 21079   19849   1737       1626
##  7 Australia 1996-01-01  10.6  18311    0.237 21923   21079   1846       1737
##  8 Australia 1997-01-01  10.3  18518    0.239 22961   21923   1948       1846
##  9 Australia 1998-01-01  10.5  18711    0.242 24148   22961   2077       1948
## 10 Australia 1999-01-01   8.67 18926    0.244 25445   24148   2231       2077
## # … with 18 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

This could get cumbersome fast.

Use %in% for multiple selections

```r
my_countries <- c("Australia", "Canada", "United States", "Ireland")

organdata %>%
  filter(country %in% my_countries)
```

```
## # A tibble: 56 x 21
##    country   year       donors   pop pop_dens   gdp gdp_lag health health_lag
##    <chr>     <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Australia NA          NA    17065    0.220 16774   16591   1300       1224
##  2 Australia 1991-01-01  12.1  17284    0.223 17171   16774   1379       1300
##  3 Australia 1992-01-01  12.4  17495    0.226 17914   17171   1455       1379
##  4 Australia 1993-01-01  12.5  17667    0.228 18883   17914   1540       1455
##  5 Australia 1994-01-01  10.2  17855    0.231 19849   18883   1626       1540
##  6 Australia 1995-01-01  10.2  18072    0.233 21079   19849   1737       1626
##  7 Australia 1996-01-01  10.6  18311    0.237 21923   21079   1846       1737
##  8 Australia 1997-01-01  10.3  18518    0.239 22961   21923   1948       1846
##  9 Australia 1998-01-01  10.5  18711    0.242 24148   22961   2077       1948
## 10 Australia 1999-01-01   8.67 18926    0.244 25445   24148   2231       2077
## # … with 46 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

Negating %in%

```r
my_countries <- c("Australia", "Canada", "United States", "Ireland")

organdata %>%
  filter(!(country %in% my_countries))
```

```
## # A tibble: 182 x 21
##    country year       donors   pop pop_dens   gdp gdp_lag health health_lag
##    <chr>   <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Austria NA           NA    7678     9.16 18914   17425   1344       1255
##  2 Austria 1991-01-01   27.6  7755     9.25 19860   18914   1419       1344
##  3 Austria 1992-01-01   23.1  7841     9.35 20601   19860   1551       1419
##  4 Austria 1993-01-01   26.2  7906     9.43 21119   20601   1674       1551
##  5 Austria 1994-01-01   21.4  7936     9.46 21940   21119   1739       1674
##  6 Austria 1995-01-01   21.5  7948     9.48 22817   21940   1865       1739
##  7 Austria 1996-01-01   24.7  7959     9.49 23798   22817   1986       1865
##  8 Austria 1997-01-01   19.5  7968     9.50 24364   23798   1848       1986
##  9 Austria 1998-01-01   20.7  7977     9.51 25423   24364   1953       1848
## 10 Austria 1999-01-01   25.9  7992     9.53 26513   25423   2069       1953
## # … with 172 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

Also a bit awkward. There's no built-in "Not in" operator.

Negating `%in%`

We can make one!

```
`%nin%` <- Negate(`%in%`) # this operator is included in the socviz package
```

(The backticks are special here because we need to name an operator.)

Negating `%in%`

We can make one!

```
`%nin%` <- Negate(`%in%`) # this operator is included in the socviz package
```

(The backticks are special here because we need to name an operator.)

```
organdata %>%
  filter(country %nin% my_countries)
```

```
## # A tibble: 182 x 21
##    country year       donors   pop pop_dens   gdp gdp_lag health health_lag
##    <chr>   <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Austria NA           NA    7678     9.16 18914   17425   1344       1255
##  2 Austria 1991-01-01   27.6  7755     9.25 19860   18914   1419       1344
##  3 Austria 1992-01-01   23.1  7841     9.35 20601   19860   1551       1419
##  4 Austria 1993-01-01   26.2  7906     9.43 21119   20601   1674       1551
##  5 Austria 1994-01-01   21.4  7936     9.46 21940   21119   1739       1674
##  6 Austria 1995-01-01   21.5  7948     9.48 22817   21940   1865       1739
##  7 Austria 1996-01-01   24.7  7959     9.49 23798   22817   1986       1865
##  8 Austria 1997-01-01   19.5  7968     9.50 24364   23798   1848       1986
##  9 Austria 1998-01-01   20.7  7977     9.51 25423   24364   1953       1848
## 10 Austria 1999-01-01   25.9  7992     9.53 26513   25423   2069       1953
## # … with 172 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

Doing more than one thing

Earlier we saw this:

```r
gss_sm %>%
  group_by(race, sex, degree) %>%
  summarize(n = n(),
            mean_age = mean(age, na.rm = TRUE),
            mean_kids = mean(childs, na.rm = TRUE))
```

```
## # A tibble: 34 x 6
## # Groups:   race, sex [6]
##    race  sex    degree             n mean_age mean_kids
##    <fct> <fct>  <fct>          <int>    <dbl>     <dbl>
##  1 White Male   Lt High School    96     52.9      2.45
##  2 White Male   High School      470     48.8      1.61
##  3 White Male   Junior College    65     47.1      1.54
##  4 White Male   Bachelor         208     48.6      1.35
##  5 White Male   Graduate         112     56.0      1.71
##  6 White Female Lt High School   101     55.4      2.81
##  7 White Female High School      587     51.9      1.98
##  8 White Female Junior College   101     48.2      1.91
##  9 White Female Bachelor         218     49.2      1.44
## 10 White Female Graduate         138     53.6      1.38
## # … with 24 more rows
```

Doing more than one thing

Similarly for `organdata` we might want to do:

```
organdata %>%
  group_by(consent_law, country) %>%
  summarize(donors_mean = mean(donors, na.rm = TRUE),
            donors_sd = sd(donors, na.rm = TRUE),
            gdp_mean = mean(gdp, na.rm = TRUE),
            health_mean = mean(health, na.rm = TRUE),
            roads_mean = mean(roads, na.rm = TRUE))
```

```
## # A tibble: 17 x 7
## # Groups:   consent_law [2]
##    consent_law country     donors_mean donors_sd gdp_mean health_mean roads_mean
##    <chr>       <chr>             <dbl>     <dbl>    <dbl>       <dbl>      <dbl>
##  1 Informed    Australia          10.6     1.14    22179.       1958.      105. 
##  2 Informed    Canada             14.0     0.751   23711.       2272.      109. 
##  3 Informed    Denmark            13.1     1.47    23722.       2054.      102. 
##  4 Informed    Germany            13.0     0.611   22163.       2349.      113. 
##  5 Informed    Ireland            19.8     2.48    20824.       1480.      118. 
##  6 Informed    Netherlands        13.7     1.55    23013.       1993.       76.1
##  7 Informed    United Kin…        13.5     0.775   21359.       1561.       67.9
##  8 Informed    United Sta…        20.0     1.33    29212.       3988.      155. 
##  9 Presumed    Austria            23.5     2.42    23876.       1875.      150. 
## 10 Presumed    Belgium            21.9     1.94    22500.       1958.      155. 
## 11 Presumed    Finland            18.4     1.53    21019.       1615.       93.6
## 12 Presumed    France             16.8     1.60    22603.       2160.      156. 
## 13 Presumed    Italy              11.1     4.28    21554.       1757       122. 
## 14 Presumed    Norway             15.4     1.11    26448.       2217.       70.0
## 15 Presumed    Spain              28.1     4.96    16933        1289.      161. 
## 16 Presumed    Sweden             13.1     1.75    22415.       1951.       72.3
## 17 Presumed    Switzerland        14.2     1.71    27233        2776.       96.4
```

This works, but it's really tedious. Also error-prone.

Doing more than one thing with `across()`

Instead, use `across()` to apply a function to more than one column.

```
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(avg = mean),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 5
## # Groups:   consent_law [2]
##    consent_law country        gdp_avg donors_avg roads_avg
##    <chr>       <chr>            <dbl>      <dbl>     <dbl>
##  1 Informed    Australia       22179.       10.6     105.
##  2 Informed    Canada          23711.       14.0     109.
##  3 Informed    Denmark         23722.       13.1     102.
##  4 Informed    Germany         22163.       13.0     113.
##  5 Informed    Ireland         20824.       19.8     118.
##  6 Informed    Netherlands     23013.       13.7      76.1
##  7 Informed    United Kingdom  21359.       13.5      67.9
##  8 Informed    United States   29212.       20.0     155.
##  9 Presumed    Austria         23876.       23.5     150.
## 10 Presumed    Belgium         22500.       21.9     155.
## 11 Presumed    Finland         21019.       18.4      93.6
## 12 Presumed    France          22603.       16.8     156.
## 13 Presumed    Italy           21554.       11.1     122.
## 14 Presumed    Norway          26448.       15.4      70.0
## 15 Presumed    Spain           16933        28.1     161.
## 16 Presumed    Sweden          22415.       13.1      72.3
```

Let's look at that again

```r
my_vars <- c("gdp", "donors", "roads")
```

Let's look at that again

```r
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata
```

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

Let's look at that again

```r
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata %>%
  group_by(consent_law, country)
```

```
## # A tibble: 238 x 21
## # Groups:   consent_law, country [17]
##    country   year       donors   pop pop_dens   gdp gdp_lag health health_lag
##    <chr>     <date>      <dbl> <int>    <dbl> <int>   <int>  <dbl>      <dbl>
##  1 Australia NA          NA    17065    0.220 16774   16591   1300       1224
##  2 Australia 1991-01-01  12.1  17284    0.223 17171   16774   1379       1300
##  3 Australia 1992-01-01  12.4  17495    0.226 17914   17171   1455       1379
##  4 Australia 1993-01-01  12.5  17667    0.228 18883   17914   1540       1455
##  5 Australia 1994-01-01  10.2  17855    0.231 19849   18883   1626       1540
##  6 Australia 1995-01-01  10.2  18072    0.233 21079   19849   1737       1626
##  7 Australia 1996-01-01  10.6  18311    0.237 21923   21079   1846       1737
##  8 Australia 1997-01-01  10.3  18518    0.239 22961   21923   1948       1846
##  9 Australia 1998-01-01  10.5  18711    0.242 24148   22961   2077       1948
## 10 Australia 1999-01-01   8.67 18926    0.244 25445   24148   2231       2077
## # … with 228 more rows, and 12 more variables: pubhealth <dbl>, roads <dbl>,
## #   cerebvas <int>, assault <int>, external <int>, txp_pop <dbl>, world <chr>,
## #   opt <chr>, consent_law <chr>, consent_practice <chr>, consistent <chr>,
## #   ccode <chr>
```

Let's look at that again

```r
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(avg = mean),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 5
## # Groups:   consent_law [2]
##    consent_law country        gdp_avg donors_avg roads_avg
##    <chr>       <chr>            <dbl>      <dbl>     <dbl>
##  1 Informed    Australia       22179.       10.6     105. 
##  2 Informed    Canada          23711.       14.0     109. 
##  3 Informed    Denmark         23722.       13.1     102. 
##  4 Informed    Germany         22163.       13.0     113. 
##  5 Informed    Ireland         20824.       19.8     118. 
##  6 Informed    Netherlands     23013.       13.7      76.1
##  7 Informed    United Kingdom  21359.       13.5      67.9
##  8 Informed    United States   29212.       20.0     155. 
##  9 Presumed    Austria         23876.       23.5     150. 
## 10 Presumed    Belgium         22500.       21.9     155. 
## 11 Presumed    Finland         21019.       18.4      93.6
## 12 Presumed    France          22603.       16.8     156. 
## 13 Presumed    Italy           21554.       11.1     122. 
## 14 Presumed    Norway          26448.       15.4      70.0
## 15 Presumed    Spain           16933        28.1     161. 
## 16 Presumed    Sweden          22415.       13.1      72.3
## 17 Presumed    Switzerland     27233        14.2      96.4
```

`my_vars` are selected by `across()`

Let's look at that again

```
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(avg = mean),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 5
## # Groups:   consent_law [2]
##    consent_law country        gdp_avg donors_avg roads_avg
##    <chr>       <chr>            <dbl>      <dbl>     <dbl>
##  1 Informed    Australia       22179.       10.6     105.
##  2 Informed    Canada          23711.       14.0     109.
##  3 Informed    Denmark         23722.       13.1     102.
##  4 Informed    Germany         22163.       13.0     113.
##  5 Informed    Ireland         20824.       19.8     118.
##  6 Informed    Netherlands     23013.       13.7      76.1
##  7 Informed    United Kingdom  21359.       13.5      67.9
##  8 Informed    United States   29212.       20.0     155.
##  9 Presumed    Austria         23876.       23.5     150.
## 10 Presumed    Belgium         22500.       21.9     155.
## 11 Presumed    Finland         21019.       18.4      93.6
## 12 Presumed    France          22603.       16.8     156.
## 13 Presumed    Italy           21554.       11.1     122.
## 14 Presumed    Norway          26448.       15.4      70.0
## 15 Presumed    Spain           16933        28.1     161.
## 16 Presumed    Sweden          22415.       13.1      72.3
## 17 Presumed    Switzerland     27233        14.2      96.4
```

`my_vars` are selected by `across()`

`list()` of the form `result = function` gives the new columns that will be calculated.

Let's look at that again

```r
my_vars <- c("gdp", "donors", "roads")

## nested parens again, but it's worth it
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(avg = mean),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 5
## # Groups:   consent_law [2]
##    consent_law country        gdp_avg donors_avg roads_avg
##    <chr>       <chr>            <dbl>      <dbl>     <dbl>
##  1 Informed    Australia       22179.       10.6     105. 
##  2 Informed    Canada          23711.       14.0     109. 
##  3 Informed    Denmark         23722.       13.1     102. 
##  4 Informed    Germany         22163.       13.0     113. 
##  5 Informed    Ireland         20824.       19.8     118. 
##  6 Informed    Netherlands     23013.       13.7      76.1
##  7 Informed    United Kingdom  21359.       13.5      67.9
##  8 Informed    United States   29212.       20.0     155. 
##  9 Presumed    Austria         23876.       23.5     150. 
## 10 Presumed    Belgium         22500.       21.9     155. 
## 11 Presumed    Finland         21019.       18.4      93.6
## 12 Presumed    France          22603.       16.8     156. 
## 13 Presumed    Italy           21554.       11.1     122. 
## 14 Presumed    Norway          26448.       15.4      70.0
## 15 Presumed    Spain           16933        28.1     161. 
## 16 Presumed    Sweden          22415.       13.1      72.3
## 17 Presumed    Switzerland     27233        14.2      96.4
```

`my_vars` are selected by `across()`

`list()` of the form `result = function` gives the new columns that will be calculated.

`na.rm = TRUE` is passed through to the functions inside the `list()`

We can calculate more than one thing

```r
my_vars <- c("gdp", "donors", "roads")

organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(avg = mean,
                        sd = var,
                        md = median),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 11
## # Groups:   consent_law [2]
##    consent_law country     gdp_avg  gdp_sd gdp_md donors_avg donors_sd donors_md
##    <chr>       <chr>         <dbl>   <dbl>  <int>      <dbl>     <dbl>     <dbl>
##  1 Informed    Australia    22179. 1.57e7 21923       10.6     1.31       10.4
##  2 Informed    Canada       23711. 1.57e7 22764       14.0     0.564      14.0
##  3 Informed    Denmark      23722. 1.52e7 23548       13.1     2.16       12.9
##  4 Informed    Germany      22163. 6.26e6 22164       13.0     0.374      13
##  5 Informed    Ireland      20824. 4.45e7 19245       19.8     6.14       19.2
##  6 Informed    Netherlands  23013. 1.42e7 22541       13.7     2.41       13.8
##  7 Informed    United Kin…  21359. 1.54e7 20839       13.5     0.601      13.5
##  8 Informed    United Sta…  29212. 2.09e7 28772       20.0     1.76       20.1
##  9 Presumed    Austria      23876. 1.12e7 23798       23.5     5.84       23.8
## 10 Presumed    Belgium      22500. 1.01e7 22152       21.9     3.75       21.4
## 11 Presumed    Finland      21019. 1.35e7 19842       18.4     2.33       19.4
## 12 Presumed    France       22603. 1.06e7 21990       16.8     2.55       16.6
## 13 Presumed    Italy        21554. 7.74e6 21396       11.1    18.3        11.3
## 14 Presumed    Norway       26448. 4.21e7 26218       15.4     1.23       15.4
## 15 Presumed    Spain        16933  8.34e6 16416       28.1    24.6        28
## 16 Presumed    Sweden       22415. 1.03e7 22029       13.1     3.07       12.7
## 17 Presumed    Switzerland 27233   4.64e6 26304       14.2     2.92       14.4
## # … with 3 more variables: roads_avg <dbl>, roads_sd <dbl>, roads_md <dbl>
```

It's OK to use the function names

```r
my_vars <- c("gdp", "donors", "roads")

organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(my_vars,
                   list(mean = mean,
                        var = var,
                        median = median),
                   na.rm = TRUE))
```

```
## # A tibble: 17 x 11
## # Groups:   consent_law [2]
##    consent_law country        gdp_mean  gdp_var gdp_median donors_mean donors_var
##    <chr>       <chr>             <dbl>    <dbl>      <int>       <dbl>      <dbl>
##  1 Informed    Australia        22179.   1.57e7      21923        10.6      1.31 
##  2 Informed    Canada           23711.   1.57e7      22764        14.0      0.564
##  3 Informed    Denmark          23722.   1.52e7      23548        13.1      2.16 
##  4 Informed    Germany          22163.   6.26e6      22164        13.0      0.374
##  5 Informed    Ireland          20824.   4.45e7      19245        19.8      6.14 
##  6 Informed    Netherlands      23013.   1.42e7      22541        13.7      2.41 
##  7 Informed    United Kingd…    21359.   1.54e7      20839        13.5      0.601
##  8 Informed    United States    29212.   2.09e7      28772        20.0      1.76 
##  9 Presumed    Austria          23876.   1.12e7      23798        23.5      5.84 
## 10 Presumed    Belgium          22500.   1.01e7      22152        21.9      3.75 
## 11 Presumed    Finland          21019.   1.35e7      19842        18.4      2.33 
## 12 Presumed    France           22603.   1.06e7      21990        16.8      2.55 
## 13 Presumed    Italy            21554.   7.74e6      21396        11.1     18.3  
## 14 Presumed    Norway           26448.   4.21e7      26218        15.4      1.23 
## 15 Presumed    Spain            16933    8.34e6      16416        28.1     24.6  
## 16 Presumed    Sweden           22415.   1.03e7      22029        13.1      3.07 
## 17 Presumed    Switzerland      27233    4.64e6      26304        14.2      2.92 
## # … with 4 more variables: donors_median <dbl>, roads_mean <dbl>,
## #   roads_var <dbl>, roads_median <dbl>
```

Conditionally select with `across(where())`

```r
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(where(is.numeric),
                   list(mean = mean,
                        var = var,
                        median = median),
                   na.rm = TRUE)) %>%
    print(n = 3) # just to save slide space
```

```
## # A tibble: 17 x 41
## # Groups:   consent_law [2]
##   consent_law country   donors_mean donors_var donors_median pop_mean  pop_var
##   <chr>       <chr>           <dbl>      <dbl>         <dbl>    <dbl>    <dbl>
## 1 Informed    Australia        10.6      1.31           10.4   18318.  690385.
## 2 Informed    Canada           14.0      0.564          14.0   29608. 1422648.
## 3 Informed    Denmark          13.1      2.16           12.9    5257.    6497.
## # … with 14 more rows, and 34 more variables: pop_median <int>,
## #   pop_dens_mean <dbl>, pop_dens_var <dbl>, pop_dens_median <dbl>,
## #   gdp_mean <dbl>, gdp_var <dbl>, gdp_median <int>, gdp_lag_mean <dbl>,
## #   gdp_lag_var <dbl>, gdp_lag_median <dbl>, health_mean <dbl>,
## #   health_var <dbl>, health_median <dbl>, health_lag_mean <dbl>,
## #   health_lag_var <dbl>, health_lag_median <dbl>, pubhealth_mean <dbl>,
## #   pubhealth_var <dbl>, pubhealth_median <dbl>, roads_mean <dbl>,
## #   roads_var <dbl>, roads_median <dbl>, cerebvas_mean <dbl>,
## #   cerebvas_var <dbl>, cerebvas_median <int>, assault_mean <dbl>,
## #   assault_var <dbl>, assault_median <int>, external_mean <dbl>,
## #   external_var <dbl>, external_median <int>, txp_pop_mean <dbl>,
## #   txp_pop_var <dbl>, txp_pop_median <dbl>
```

Name new columns with `.names`

```
organdata %>%
  group_by(consent_law, country) %>%
  summarize(across(where(is.numeric),
                   list(mean = mean,
                        var = var,
                        median = median),
                   na.rm = TRUE,
                   .names = "{fn}_{col}")) %>%
  print(n = 3)
```

```
## # A tibble: 17 x 41
## # Groups:   consent_law [2]
##   consent_law country   mean_donors var_donors median_donors mean_pop  var_pop
##   <chr>       <chr>           <dbl>      <dbl>         <dbl>    <dbl>    <dbl>
## 1 Informed    Australia        10.6      1.31           10.4   18318.  690385.
## 2 Informed    Canada           14.0      0.564          14.0   29608. 1422648.
## 3 Informed    Denmark          13.1      2.16           12.9    5257.    6497.
## # … with 14 more rows, and 34 more variables: median_pop <int>,
## #   mean_pop_dens <dbl>, var_pop_dens <dbl>, median_pop_dens <dbl>,
## #   mean_gdp <dbl>, var_gdp <dbl>, median_gdp <int>, mean_gdp_lag <dbl>,
## #   var_gdp_lag <dbl>, median_gdp_lag <dbl>, mean_health <dbl>,
## #   var_health <dbl>, median_health <dbl>, mean_health_lag <dbl>,
## #   var_health_lag <dbl>, median_health_lag <dbl>, mean_pubhealth <dbl>,
## #   var_pubhealth <dbl>, median_pubhealth <dbl>, mean_roads <dbl>,
## #   var_roads <dbl>, median_roads <dbl>, mean_cerebvas <dbl>,
## #   var_cerebvas <dbl>, median_cerebvas <int>, mean_assault <dbl>,
## #   var_assault <dbl>, median_assault <int>, mean_external <dbl>,
## #   var_external <dbl>, median_external <int>, mean_txp_pop <dbl>,
## #   var_txp_pop <dbl>, median_txp_pop <dbl>
```

In tidyverse functions, arguments that begin with a "`.`" generally have it in order to avoid confusion with existing items, or are "pronouns" referring to e.g. "the name of the thing we're currently talking about as we evaluate this function".

This all works with `mutate()`, too

```
organdata %>%
  mutate(across(where(is.character), toupper)) %>%
  select(where(is.character))
```

```
## # A tibble: 238 x 7
##    country   world   opt   consent_law consent_practice consistent ccode
##    <chr>     <chr>   <chr> <chr>       <chr>            <chr>      <chr>
##  1 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  2 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  3 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  4 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  5 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  6 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  7 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  8 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
##  9 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
## 10 AUSTRALIA LIBERAL IN    INFORMED    INFORMED         YES        OZ
## # … with 228 more rows
```

Arrange rows and columns

Sort rows with `arrange()`

```r
organdata %>%
  group_by(consent_law, country) %>%
  summarize(donors = mean(donors, na.rm = TRUE)) %>%
  arrange(donors) %>% ##<
  print(n = 5)
```

```
## # A tibble: 17 x 3
## # Groups:   consent_law [2]
##   consent_law country   donors
##   <chr>       <chr>      <dbl>
## 1 Informed    Australia   10.6
## 2 Presumed    Italy       11.1
## 3 Informed    Germany     13.0
## 4 Informed    Denmark     13.1
## 5 Presumed    Sweden      13.1
## # … with 12 more rows
```

```r
organdata %>%
  group_by(consent_law, country) %>%
  summarize(donors = mean(donors, na.rm = TRUE)) %>%
  arrange(desc(donors)) %>%  ##<
  print(n = 5)
```

```
## # A tibble: 17 x 3
## # Groups:   consent_law [2]
##   consent_law country       donors
##   <chr>       <chr>          <dbl>
## 1 Presumed    Spain           28.1
## 2 Presumed    Austria         23.5
## 3 Presumed    Belgium         21.9
## 4 Informed    United States   20.0
## 5 Informed    Ireland         19.8
## # … with 12 more rows
```

Using `arrange()` to order rows in this way won't respect groupings.

More generally ...

```
organdata %>%
  group_by(consent_law, country) %>%
  summarize(donors = mean(donors, na.rm = TRUE)) %>%
  slice_max(donors, n = 5)
```

```
## # A tibble: 10 x 3
## # Groups:   consent_law [2]
##    consent_law country        donors
##    <chr>       <chr>           <dbl>
##  1 Informed    United States    20.0
##  2 Informed    Ireland          19.8
##  3 Informed    Canada           14.0
##  4 Informed    Netherlands      13.7
##  5 Informed    United Kingdom   13.5
##  6 Presumed    Spain            28.1
##  7 Presumed    Austria          23.5
##  8 Presumed    Belgium          21.9
##  9 Presumed    Finland          18.4
## 10 Presumed    France           16.8
```

You can see that `slice_max()` respects grouping.

There's `slice_min()`, `.slice_head()`, `slice_tail()`, `slice_sample()`, and the most general one, `slice()`.

dplyr's window functions

Ranking and cumulation within groups.

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

covnat_weekly
```

```
## # A tibble: 12,746 x 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   Afghanist… AFG    3.89e7     0      0        0         0
##  2 2020-01-06 2020-02   Afghanist… AFG    3.89e7     0      0        0         0
##  3 2020-01-13 2020-03   Afghanist… AFG    3.89e7     0      0        0         0
##  4 2020-01-20 2020-04   Afghanist… AFG    3.89e7     0      0        0         0
##  5 2020-01-27 2020-05   Afghanist… AFG    3.89e7     0      0        0         0
##  6 2020-02-03 2020-06   Afghanist… AFG    3.89e7     0      0        0         0
##  7 2020-02-10 2020-07   Afghanist… AFG    3.89e7     0      0        0         0
##  8 2020-02-17 2020-08   Afghanist… AFG    3.89e7     0      0        0         0
##  9 2020-02-24 2020-09   Afghanist… AFG    3.89e7     1      0        1         0
## 10 2020-03-02 2020-10   Afghanist… AFG    3.89e7     3      0        4         0
## # … with 12,736 more rows, and 2 more variables: r14_cases <dbl>,
## #   r14_deaths <dbl>
```

dplyr's **window** functions

`cumsum()` gives cumulative sums

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

```
## # A tibble: 66 x 5
##    date       cname         iso3  cases cumulative
##    <date>     <chr>         <chr> <dbl>      <dbl>
##  1 2019-12-30 United States USA       0          0
##  2 2020-01-06 United States USA       0          0
##  3 2020-01-13 United States USA       0          0
##  4 2020-01-20 United States USA       5          5
##  5 2020-01-27 United States USA       6         11
##  6 2020-02-03 United States USA       1         12
##  7 2020-02-10 United States USA       3         15
##  8 2020-02-17 United States USA      20         35
##  9 2020-02-24 United States USA      54         89
## 10 2020-03-02 United States USA     465        554
## # … with 56 more rows
```

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: 6 x 4
##   date       cname         iso3  deaths
##   <date>     <chr>         <chr>  <dbl>
## 1 2021-01-04 United States USA    22852
## 2 2021-01-11 United States USA    23169
## 3 2021-01-18 United States USA    23518
## 4 2021-01-25 United States USA    22226
## 5 2021-02-01 United States USA    20127
## 6 2021-02-08 United States USA    22843
```

The `dplyr` vignette on Window functions is good.

An application

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

```
## # A tibble: 371 x 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 x 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.

```r
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 x 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.

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

Tidy moving averages with `slider`

```r
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 x 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 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, 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 x 32
##     year    id ballot   age childs  sibs degree   race  sex    region  income16
##    <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>  <fct>   <fct>   
##  1  2016     1      1    47      3     2 Bachelor White Male   New En… $170000 …
##  2  2016     2      2    61      0     3 High Sc… White Male   New En… $50000 t…
##  3  2016     3      3    72      2     3 Bachelor White Male   New En… $75000 t…
##  4  2016     4      1    43      4     3 High Sc… White Female New En… $170000 …
##  5  2016     5      3    55      2     2 Graduate White Female New En… $170000 …
##  6  2016     6      2    53      2     2 Junior … White Female New En… $60000 t…
##  7  2016     7      1    50      2     2 High Sc… White Male   New En… $170000 …
##  8  2016     8      3    23      3     6 High Sc… Other Female Middle… $30000 t…
##  9  2016     9      1    45      3     5 High Sc… Black Male   Middle… $60000 t…
## 10  2016    10      3    71      4     1 Junior … White Male   Middle… $60000 t…
## # … 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 <dbl>, 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

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

```
## # A tibble: 2,867 x 19
##    region   bigregion  year    id ballot   age childs  sibs degree   race  sex
##    <fct>    <fct>     <dbl> <dbl>  <dbl> <dbl>  <dbl> <dbl> <fct>    <fct> <fct>
##  1 New Eng… Northeast  2016     1      1    47      3     2 Bachelor White Male
##  2 New Eng… Northeast  2016     2      2    61      0     3 High Sc… White Male
##  3 New Eng… Northeast  2016     3      3    72      2     3 Bachelor White Male
##  4 New Eng… Northeast  2016     4      1    43      4     3 High Sc… White Fema…
##  5 New Eng… Northeast  2016     5      3    55      2     2 Graduate 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 Sc… White Male
##  8 Middle … Northeast  2016     8      3    23      3     6 High Sc… Other Fema…
##  9 Middle … Northeast  2016     9      1    45      3     5 High Sc… 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 <dbl>, partners_rc <fct>,
## #   income16 <fct>, income_rc <fct>
```

Shuffle columns around

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

```
## # A tibble: 2,867 x 19
##    region    bigregion  year    id ballot   age children siblings degree   race
##    <fct>     <fct>     <dbl> <dbl>  <dbl> <dbl>    <dbl>    <dbl> <fct>    <fct>
##  1 New Engl… Northeast  2016     1      1    47        3        2 Bachelor White
##  2 New Engl… Northeast  2016     2      2    61        0        3 High Sc… White
##  3 New Engl… Northeast  2016     3      3    72        2        3 Bachelor White
##  4 New Engl… Northeast  2016     4      1    43        4        3 High Sc… White
##  5 New Engl… Northeast  2016     5      3    55        2        2 Graduate White
##  6 New Engl… Northeast  2016     6      2    53        2        2 Junior … White
##  7 New Engl… Northeast  2016     7      1    50        2        2 High Sc… White
##  8 Middle A… Northeast  2016     8      3    23        3        6 High Sc… Other
##  9 Middle A… Northeast  2016     9      1    45        3        5 High Sc… Black
## 10 Middle A… 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 <dbl>,
## #   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 x 19
##       id region    bigregion  year ballot   age children siblings degree   race
##    <dbl> <fct>     <fct>     <dbl>  <dbl> <dbl>    <dbl>    <dbl> <fct>    <fct>
##  1     1 New Engl… Northeast  2016      1    47        3        2 Bachelor White
##  2     2 New Engl… Northeast  2016      2    61        0        3 High Sc… White
##  3     3 New Engl… Northeast  2016      3    72        2        3 Bachelor White
##  4     4 New Engl… Northeast  2016      1    43        4        3 High Sc… White
##  5     5 New Engl… Northeast  2016      3    55        2        2 Graduate White
##  6     6 New Engl… Northeast  2016      2    53        2        2 Junior … White
##  7     7 New Engl… Northeast  2016      1    50        2        2 High Sc… White
##  8     8 Middle A… Northeast  2016      3    23        3        6 High Sc… Other
##  9     9 Middle A… Northeast  2016      1    45        3        5 High Sc… Black
## 10    10 Middle A… 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 <dbl>,
## #   partners_rc <fct>, income16 <fct>, income_rc <fct>
```

Shuffle columns around

```r
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 x 18
##       id region bigregion  year   age children siblings degree race  sex   padeg
##    <dbl> <fct>  <fct>     <dbl> <dbl>    <dbl>    <dbl> <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 <dbl>, 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 x 18
##       id  year   age children siblings pres12 region    bigregion degree   race
##    <dbl> <dbl> <dbl>    <dbl>    <dbl>  <dbl> <fct>     <fct>     <fct>    <fct>
##  1     1  2016    47        3        2      3 New Engl… Northeast Bachelor White
##  2     2  2016    61        0        3      1 New Engl… Northeast High Sc… White
##  3     3  2016    72        2        3      2 New Engl… Northeast Bachelor White
##  4     4  2016    43        4        3      2 New Engl… Northeast High Sc… White
##  5     5  2016    55        2        2      1 New Engl… Northeast Graduate White
##  6     6  2016    53        2        2      1 New Engl… Northeast Junior … White
##  7     7  2016    50        2        2     NA New Engl… Northeast High Sc… White
##  8     8  2016    23        3        6     NA Middle A… Northeast High Sc… Other
##  9     9  2016    45        3        5     NA Middle A… Northeast High Sc… Black
## 10    10  2016    71        4        1      2 Middle A… 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

```r
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 x 18
##       id  year region    bigregion   age children siblings pres12 degree   race
##    <dbl> <dbl> <fct>     <fct>     <dbl>    <dbl>    <dbl>  <dbl> <fct>    <fct>
##  1     1  2016 New Engl… Northeast    47        3        2      3 Bachelor White
##  2     2  2016 New Engl… Northeast    61        0        3      1 High Sc… White
##  3     3  2016 New Engl… Northeast    72        2        3      2 Bachelor White
##  4     4  2016 New Engl… Northeast    43        4        3      2 High Sc… White
##  5     5  2016 New Engl… Northeast    55        2        2      1 Graduate White
##  6     6  2016 New Engl… Northeast    53        2        2      1 Junior … White
##  7     7  2016 New Engl… Northeast    50        2        2     NA High Sc… White
##  8     8  2016 Middle A… Northeast    23        3        6     NA High Sc… Other
##  9     9  2016 Middle A… Northeast    45        3        5     NA High Sc… Black
## 10    10  2016 Middle A… 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 x 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:

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

```
## # A tibble: 3 x 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 x 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

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

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

Comparisons filtering on proportions

So, beware.

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

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

Better:

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

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

Zero Counts in dplyr

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

df
```

```
## # A tibble: 280 x 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))
```

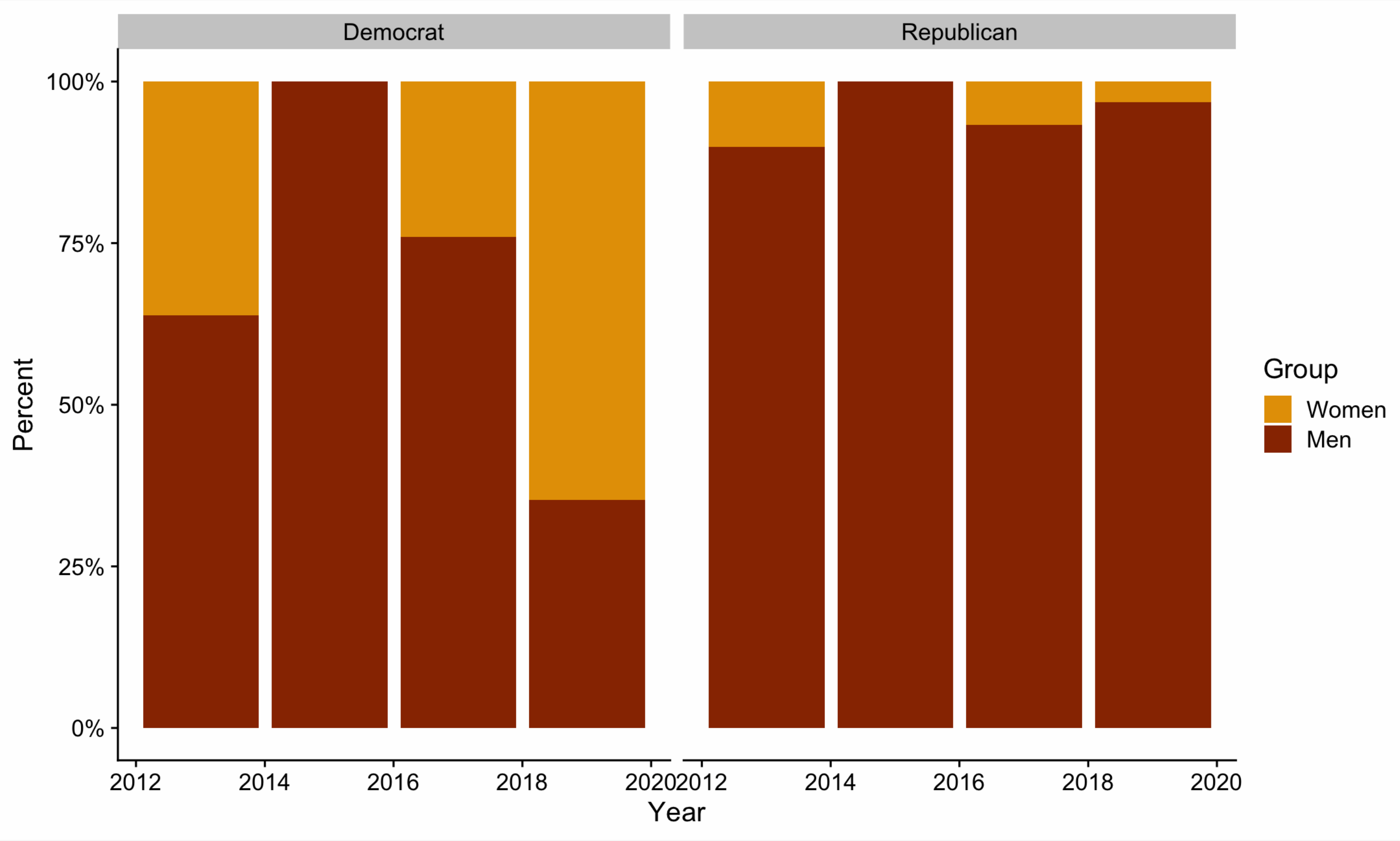
```
## # A tibble: 14 x 5
## # Groups:   start_year, party [8]
##    start_year party      sex       N   freq
##    <date>     <chr>      <chr> <int>  <dbl>
##  1 2013-01-03 Democrat   F        21 0.362
##  2 2013-01-03 Democrat   M        37 0.638
##  3 2013-01-03 Republican F         8 0.101
##  4 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

```r
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
```

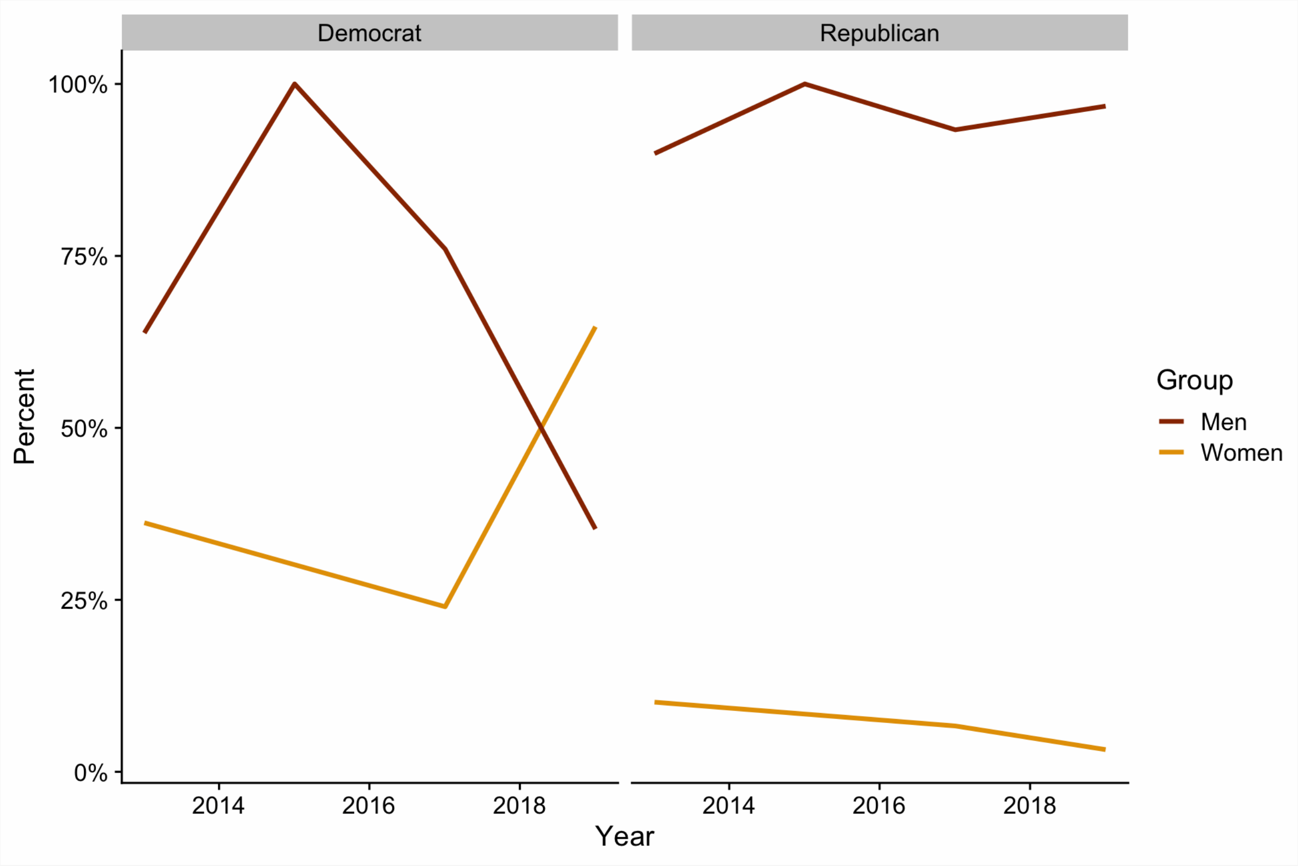


Zero Counts in dplyr

```r
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 x 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 x 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 x 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 x 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:

```r
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 x 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.

```r
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
```

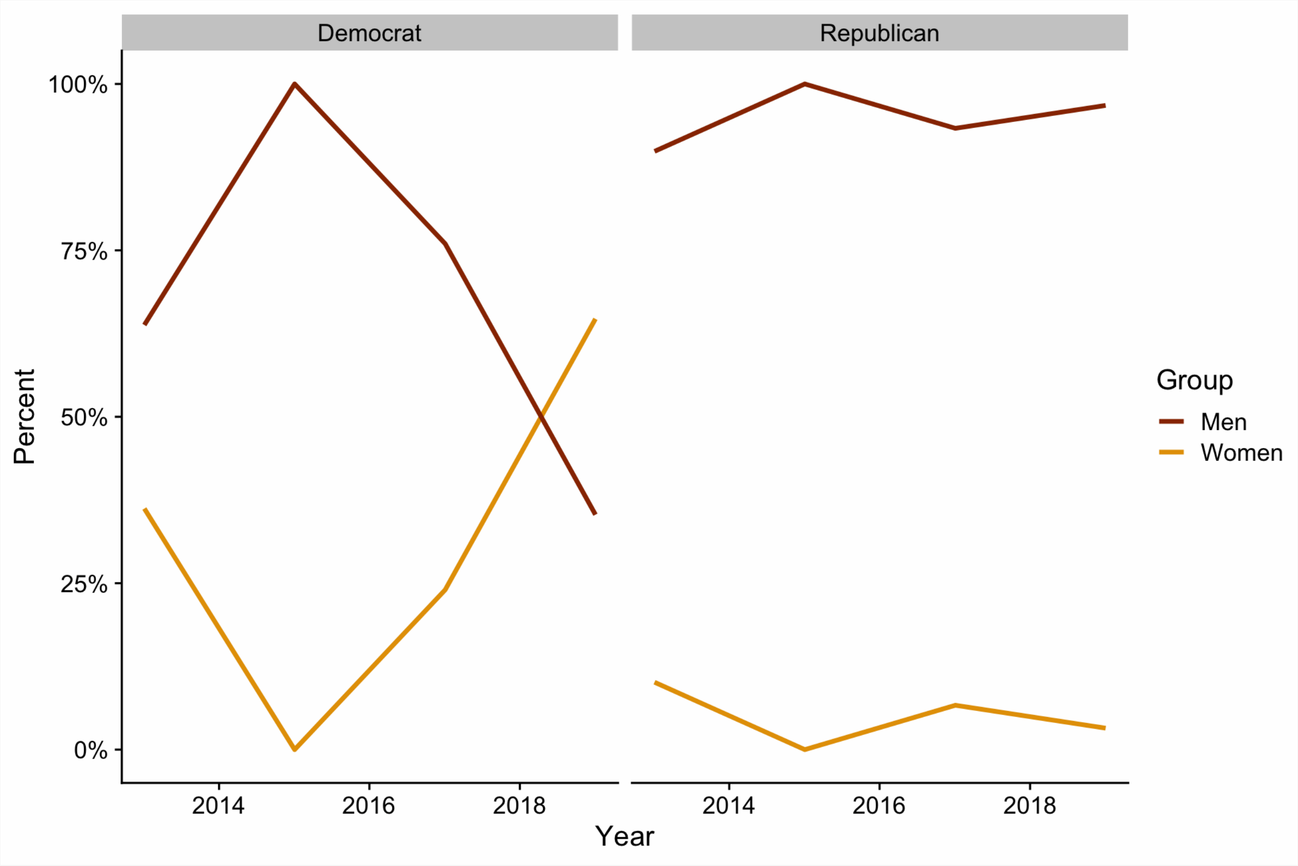
```
## # A tibble: 16 x 5
##    start_year party      sex       N   freq
##    <date>     <chr>      <chr> <dbl>  <dbl>
##  1 2013-01-03 Democrat   F        21 0.362
##  2 2013-01-03 Democrat   M        37 0.638
##  3 2013-01-03 Republican F         8 0.101
##  4 2013-01-03 Republican M        71 0.899
##  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
```


Democrat
Republican
Percent
Year
Group
Men
Women