

Manipulating Tables with **dplyr**

Data Wrangling: Session 3

Kieran Healy

Statistical Horizons, December 2022

**Time to
play with
some data**

Time to play with some **data**

woohoo!

Load our libraries

```
library(here)      # manage file paths  
library(socviz)    # data and some useful functions  
library(tidyverse) # your friend and mine
```

Tidyverse components, again

```
library(tidyverse)  
Loading tidyverse: ggplot2  
Loading tidyverse: tibble  
Loading tidyverse: tidyr  
Loading tidyverse: readr  
Loading tidyverse: purrr  
Loading tidyverse: dplyr
```

Tidyverse components, again

```
library(tidyverse)
```

```
Loading tidyverse: ggplot2
```

```
Loading tidyverse: tibble
```

```
Loading tidyverse: tidyr
```

```
Loading tidyverse: readr
```

```
Loading tidyverse: purrr
```

```
Loading tidyverse: dplyr
```

Call the package and ...

<| **Draw graphs**

<| **Nicer data tables**

<| **Tidy your data**

<| **Get data into R**

<| **Fancy Iteration**

<| **Action verbs for tables**

Other tidyverse components

forcats
haven
lubridate
readxl
stringr
reprex

Other tidyverse components

<code>forcats</code>	< Deal with factors
<code>haven</code>	< Import Stata, SPSS, etc
<code>lubridate</code>	< Dates, Durations, Times
<code>readxl</code>	< Import from spreadsheets
<code>stringr</code>	< Strings and Regular Expressions
<code>reprex</code>	< Make reproducible examples

Other tidyverse components

<code>forcats</code>	< Deal with factors
<code>haven</code>	< Import Stata, SPSS, etc
<code>lubridate</code>	< Dates, Durations, Times
<code>readxl</code>	< Import from spreadsheets
<code>stringr</code>	< Strings and Regular Expressions
<code>reprex</code>	< Make reproducible examples

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 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*”.

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. It never changes 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** is implemented by a **function**

Each **action** is implemented by a **function**

Group using `group_by()`.

Each **action** is implemented by a **function**

Group using `group_by()`.

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

Each **action** is implemented by a **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** is implemented by a **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

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

```
## # A tibble: 2,867 × 32
##   year    id ballot  age childs sibs degree race  sex  region incom...1 relig
##   <dbl> <dbl> <labe> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2    Bache... White Male  New E... $17000... None
## 2  2016     2 2      61     0 3    High ... White Male  New E... $50000... None
## 3  2016     3 3      72     2 3    Bache... White Male  New E... $75000... Cath...
## 4  2016     4 1      43     4 3    High ... White Fema... New E... $17000... Cath...
## 5  2016     5 3      55     2 2    Gradu... White Fema... New E... $17000... None
## 6  2016     6 2      53     2 2    Junio... White Fema... New E... $60000... None
## 7  2016     7 1      50     2 2    High ... White Male  New E... $17000... None
## 8  2016     8 3      23     3 6    High ... Other Fema... Middl... $30000... Cath...
## 9  2016     9 1      45     3 5    High ... Black Male  Middl... $60000... Prot...
## 10 2016    10 3      71     4 1    Junio... White Male  Middl... $60000... None
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

General Social Survey data: gss_sm

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

```
## # A tibble: 2,867 × 32
##   year    id ballot  age childs sibs degree race  sex  region incom...1 relig
##   <dbl> <dbl> <labe> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2    Bache... White Male  New E... $17000... None
## 2  2016     2 2      61     0 3    High ... White Male  New E... $50000... None
## 3  2016     3 3      72     2 3    Bache... White Male  New E... $75000... Cath...
## 4  2016     4 1      43     4 3    High ... White Fema... New E... $17000... Cath...
## 5  2016     5 3      55     2 2    Gradu... White Fema... New E... $17000... None
## 6  2016     6 2      53     2 2    Junio... White Fema... New E... $60000... None
## 7  2016     7 1      50     2 2    High ... White Male  New E... $17000... None
## 8  2016     8 3      23     3 6    High ... Other Fema... Middl... $30000... Cath...
## 9  2016     9 1      45     3 5    High ... Black Male  Middl... $60000... Prot...
## 10 2016    10 3      71     4 1    Junio... White Male  Middl... $60000... None
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

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 × 3
##       id bigregion religion
##   <dbl> <fct>      <fct>
## 1     1 1 Northeast None
## 2     2 2 Northeast None
## 3     3 3 Northeast Catholic
## 4     4 4 Northeast Catholic
## 5     5 5 Northeast None
## 6     6 6 Northeast None
## 7     7 7 Northeast None
## 8     8 8 Northeast Catholic
## 9     9 9 Northeast Protestant
## 10    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

```
gss_sm |>
  group_by(bigregion)
```

```
## # A tibble: 2,867 × 32
## # Groups:   bigregion [4]
##   year   id ballot  age childs sibs degree race  sex  region incom...1 relig
##   <dbl> <dbl> <labs> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2   Bache... White Male  New E... $17000... None
## 2  2016     2 2      61     0 3   High ... White Male  New E... $50000... None
## 3  2016     3 3      72     2 3   Bache... White Male  New E... $75000... Cath...
## 4  2016     4 1      43     4 3   High ... White Fema... New E... $17000... Cath...
## 5  2016     5 3      55     2 2   Gradu... White Fema... New E... $17000... None
## 6  2016     6 2      53     2 2   Junio... White Fema... New E... $60000... None
## 7  2016     7 1      50     2 2   High ... White Male  New E... $17000... None
## 8  2016     8 3      23     3 6   High ... Other Fema... Middl... $30000... Cath...
## 9  2016     9 1      45     3 5   High ... Black Male  Middl... $60000... Prot...
## 10 2016    10 3      71     4 1   Junio... White Male  Middl... $60000... None
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Grouping just changes the logical structure of the tibble.

Group and summarize by *one* column

```
gss_sm ## # A tibble: 2,867 × 32
##   year   id ballot  age childs sibs degree race sex region incom...1 relig
##   <dbl> <dbl> <labs> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2  Bache... White Male New E... $17000... None
## 2  2016     2 2      61     0 3  High ... White Male New E... $50000... None
## 3  2016     3 3      72     2 3  Bache... White Male New E... $75000... Cath...
## 4  2016     4 1      43     4 3  High ... White Fema... New E... $17000... Cath...
## 5  2016     5 3      55     2 2  Gradu... White Fema... New E... $17000... None
## 6  2016     6 2      53     2 2  Junio... White Fema... New E... $60000... None
## 7  2016     7 1      50     2 2  High ... White Male New E... $17000... None
## 8  2016     8 3      23     3 6  High ... Other Fema... Middl... $30000... Cath...
## 9  2016     9 1      45     3 5  High ... Black Male Middl... $60000... Prot...
## 10 2016    10 3      71     4 1  Junio... White Male Middl... $60000... None
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Group and summarize by *one* column

```
gss_sm |>
  group_by(bigregion)

## # A tibble: 2,867 × 32
## # Groups:   bigregion [4]
##   year    id ballot  age childs sibs degree race sex  region incom...1 relig
##   <dbl> <dbl> <labs> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2   Bache... White Male  New E... $17000... None
## 2  2016     2 2      61     0 3   High ... White Male  New E... $50000... None
## 3  2016     3 3      72     2 3   Bache... White Male  New E... $75000... Cath...
## 4  2016     4 1      43     4 3   High ... White Fema... New E... $17000... Cath...
## 5  2016     5 3      55     2 2   Gradu... White Fema... New E... $17000... None
## 6  2016     6 2      53     2 2   Junio... White Fema... New E... $60000... None
## 7  2016     7 1      50     2 2   High ... White Male  New E... $17000... None
## 8  2016     8 3      23     3 6   High ... Other Fema... Middl... $30000... Cath...
## 9  2016     9 1      45     3 5   High ... Black Male  Middl... $60000... Prot...
## 10 2016    10 3      71     4 1   Junio... White Male  Middl... $60000... None
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Group and summarize by *one* column

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

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

Group and summarize by *one* column

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

```
## # A tibble: 4 × 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

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

```
## # A tibble: 4 × 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 × 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 × 32
##   year    id ballot  age childs sibs degree race  sex  region incom..
##   <dbl> <dbl> <labe> <dbl>  <dbl> <lab> <fct>  <fct> <fct> <fct> <fct>
## 1  2016     1  1      47     3  2  Bache... White Male  New E... $17000
## 2  2016     2  2      61     0  3  High ... White Male  New E... $50000
## 3  2016     3  3      72     2  3  Bache... White Male  New E... $75000
## 4  2016     4  1      43     4  3  High ... White Fema... New E... $17000
## 5  2016     5  3      55     2  2  Gradu... White Fema... New E... $17000
## 6  2016     6  2      53     2  2  Junio... White Fema... New E... $60000
## 7  2016     7  1      50     2  2  High ... White Male  New E... $17000
## 8  2016     8  3      23     3  6  High ... Other Fema... Middl... $30000
## 9  2016     9  1      45     3  5  High ... Black Male  Middl... $60000
## 10 2016    10  3      71     4  1  Junio... White Male  Middl... $60000
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```


Group and summarize by *two* columns

```
gss_sm |>
```

```
  group_by(bigregion, religion)
```

```
## # A tibble: 2,867 × 32
```

```
## # Groups:   bigregion, religion [24]
```

```
##   year    id ballot  age childs sibs degree race  sex  region incom..
```

```
##   <dbl> <dbl> <labe> <dbl>  <dbl> <lab> <fct>  <fct> <fct> <fct> <fct>
```

```
## 1  2016     1  1      47      3  2  Bache... White Male  New E... $17000
```

```
## 2  2016     2  2      61      0  3  High ... White Male  New E... $50000
```

```
## 3  2016     3  3      72      2  3  Bache... White Male  New E... $75000
```

```
## 4  2016     4  1      43      4  3  High ... White Fema... New E... $17000
```

```
## 5  2016     5  3      55      2  2  Gradu... White Fema... New E... $17000
```

```
## 6  2016     6  2      53      2  2  Junio... White Fema... New E... $60000
```

```
## 7  2016     7  1      50      2  2  High ... White Male  New E... $17000
```

```
## 8  2016     8  3      23      3  6  High ... Other Fema... Middl... $30000
```

```
## 9  2016     9  1      45      3  5  High ... Black Male  Middl... $60000
```

```
## 10 2016    10  3      71      4  1  Junio... White Male  Middl... $60000
```

```
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
```

```
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>
```

```
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
```

```
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>
```

```
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
```

```
## #   abbreviated variable name ¹income16
```

Group and summarize by *two* columns

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

```
## # A tibble: 24 × 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 × 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 × 32
##   year   id ballot age childs sibs degree race sex region incom...1
##   <dbl> <dbl> <lab> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1  1      47     3  2  Bache... White Male New E... $17000...
## 2  2016     2  2      61     0  3   High ... White Male New E... $50000...
## 3  2016     3  3      72     2  3  Bache... White Male New E... $75000...
## 4  2016     4  1      43     4  3   High ... White Fema... New E... $17000...
## 5  2016     5  3      55     2  2  Gradu... White Fema... New E... $17000...
## 6  2016     6  2      53     2  2  Junio... White Fema... New E... $60000...
## 7  2016     7  1      50     2  2   High ... White Male New E... $17000...
## 8  2016     8  3      23     3  6   High ... Other Fema... Middl... $30000...
## 9  2016     9  1      45     3  5   High ... Black Male Middl... $60000...
## 10 2016    10  3      71     4  1  Junio... White Male Middl... $60000...
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Calculate frequencies

```
gss_sm |>
```

```
  group_by(bigregion, religion)
```

```
## # A tibble: 2,867 × 32
## # Groups:   bigregion, religion [24]
##   year    id ballot  age childs sibs degree race sex region incom...1
##   <dbl> <dbl> <labe> <dbl>  <dbl> <lab> <fct>  <fct> <fct> <fct> <fct>
## 1  2016     1  1      47     3  2  Bache... White Male New E... $17000...
## 2  2016     2  2      61     0  3  High ... White Male New E... $50000...
## 3  2016     3  3      72     2  3  Bache... White Male New E... $75000...
## 4  2016     4  1      43     4  3  High ... White Fema... New E... $17000...
## 5  2016     5  3      55     2  2  Gradu... White Fema... New E... $17000...
## 6  2016     6  2      53     2  2  Junio... White Fema... New E... $60000...
## 7  2016     7  1      50     2  2  High ... White Male New E... $17000...
## 8  2016     8  3      23     3  6  High ... Other Fema... Middl... $30000...
## 9  2016     9  1      45     3  5  High ... Black Male Middl... $60000...
## 10 2016    10  3      71     4  1  Junio... White Male Middl... $60000...
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Calculate frequencies

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

```
## # A tibble: 24 × 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

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

```
## # A tibble: 24 × 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

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

```
## # A tibble: 24 × 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

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

```
## # A tibble: 24 × 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

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

```
## # A tibble: 24 × 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 assumptions forward

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

```
## # A tibble: 24 × 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 assumptions forward

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

```
## # A tibble: 24 × 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 assumptions forward

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

```
## # A tibble: 24 × 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 quite clever. See how we can immediately use **freq**, even though we are creating it in the same **mutate()** expression.

Convenience functions

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

```
## # A tibble: 24 × 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 with `n()`

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

```
## # A tibble: 24 × 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 with `n()`

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

```
## # A tibble: 24 × 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 × 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; result is grouped.

Three options for counting up rows

Do it yourself with `n()`

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

```
## # A tibble: 24 × 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 × 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; result is grouped.

use `count()`

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

```
## # A tibble: 24 × 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**

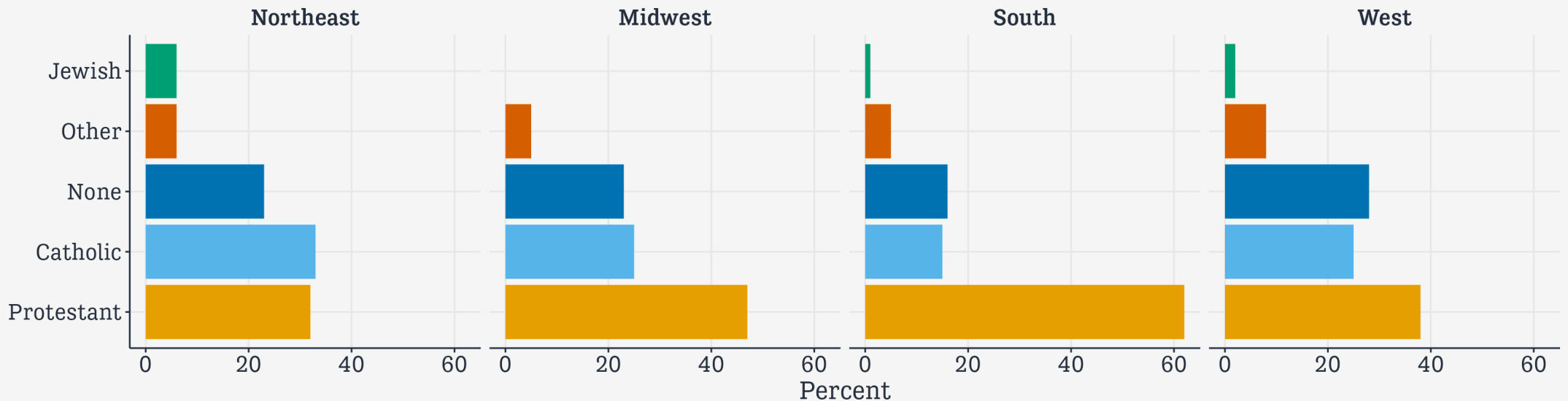
```
gss_sm |>  
  count(bigregion, religion) |>  
  pivot_wider(names_from = bigregion, values_from = n) |>  
  knitr::kable()
```

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 = "none") +
  facet_wrap(~ bigregion, nrow = 1)
```



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 × 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 × 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 × 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 assignment is a thing, 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 assignment is a thing, 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 assignment also works!

```
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 × 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 × 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
```

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

```
## # A tibble: 4 × 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?

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))
```

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

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

```
## # A tibble: 1 × 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 there are no groups carry forward to the **mutate()** step.

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

```
## # A tibble: 1 × 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 × 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.

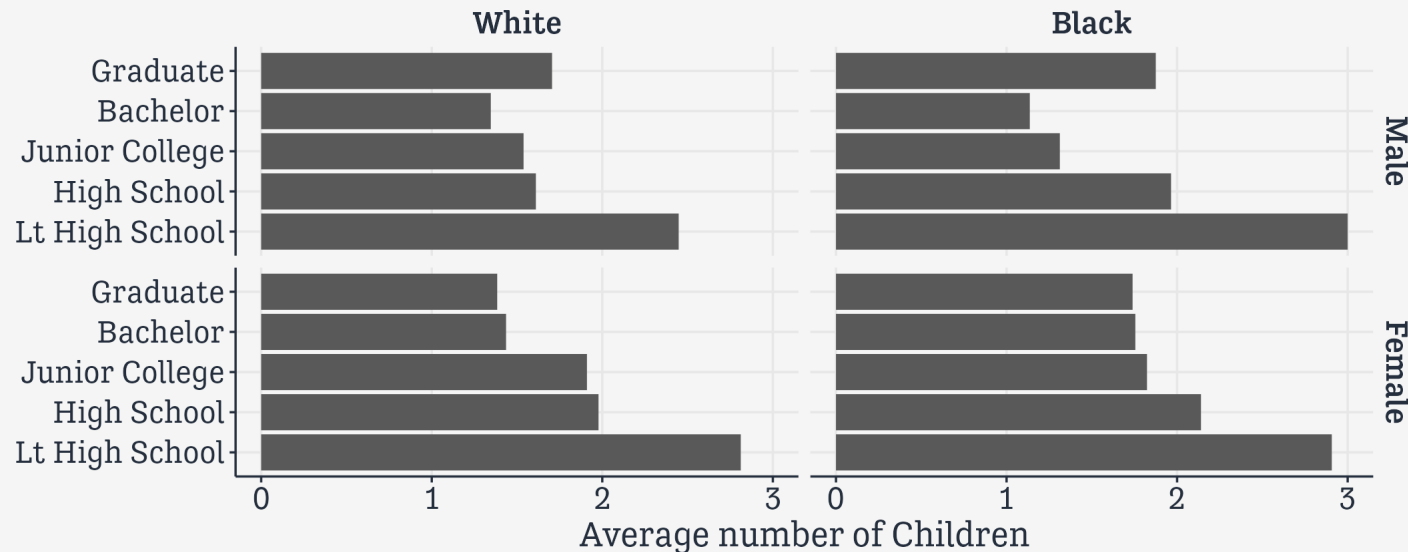
Two lessons

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(children, 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 Children", y = NULL)
```



Following a pipeline

gss_sm

```
## # A tibble: 2,867 × 32
##   year    id ballot  age childs sibs degree race sex region incom...1
##   <dbl> <dbl> <labs> <dbl> <dbl> <lab> <fct> <fct> <fct> <fct> <fct> <fct>
## 1  2016     1 1      47     3 2  Bache... White Male New E... $17000...
## 2  2016     2 2      61     0 3  High ... White Male New E... $50000...
## 3  2016     3 3      72     2 3  Bache... White Male New E... $75000...
## 4  2016     4 1      43     4 3  High ... White Fema... New E... $17000...
## 5  2016     5 3      55     2 2  Gradu... White Fema... New E... $17000...
## 6  2016     6 2      53     2 2  Junio... White Fema... New E... $60000...
## 7  2016     7 1      50     2 2  High ... White Male New E... $17000...
## 8  2016     8 3      23     3 6  High ... Other Fema... Middl... $30000...
## 9  2016     9 1      45     3 5  High ... Black Male Middl... $60000...
## 10 2016    10 3      71     4 1  Junio... White Male Middl... $60000...
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and
## #   abbreviated variable name 1income16
```

Following a pipeline

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

```
## # A tibble: 2,867 × 32  
## # Groups:   race, sex, degree [34]  
##   year    id ballot  age childs sibs  degree race  sex  region incom...1   
##   <dbl> <dbl> <labe> <dbl>  <dbl> <lab> <fct>  <fct> <fct> <fct> <fct> <fct>  
## 1  2016     1  1      47      3  2    Bache... White Male  New E... $17000... M  
## 2  2016     2  2      61      0  3    High ... White Male  New E... $50000... M  
## 3  2016     3  3      72      2  3    Bache... White Male  New E... $75000... C  
## 4  2016     4  1      43      4  3    High ... White Fema... New E... $17000... C  
## 5  2016     5  3      55      2  2    Gradu... White Fema... New E... $17000... M  
## 6  2016     6  2      53      2  2    Junio... White Fema... New E... $60000... M  
## 7  2016     7  1      50      2  2    High ... White Male  New E... $17000... M  
## 8  2016     8  3      23      3  6    High ... Other Fema... Middl... $30000... C  
## 9  2016     9  1      45      3  5    High ... Black Male  Middl... $60000... F  
## 10 2016    10  3      71      4  1    Junio... White Male  Middl... $60000... M  
## # ... with 2,857 more rows, 20 more variables: marital <fct>, padeg <fct>,  
## #   madeg <fct>, partyid <fct>, polviews <fct>, happy <fct>, partners <fct>,  
## #   grass <fct>, zodiac <fct>, pres12 <labelled>, wtssall <dbl>,  
## #   income_rc <fct>, agegrp <fct>, ageq <fct>, siblings <fct>, kids <fct>,  
## #   religion <fct>, bigregion <fct>, partners_rc <fct>, obama <dbl>, and  
## #   abbreviated variable name 1income16
```

Following a pipeline

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

```
## # A tibble: 34 × 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

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

```
## # A tibble: 34 × 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(children, na.rm = TRUE),
  mutate(pct = n/sum(n)*100) |>
  filter(race != "Other")
```

```
## # A tibble: 23 × 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

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

```
## # A tibble: 20 × 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(children, na.rm = TRUE),
            pct = n/sum(n)*100) |>
  filter(race != "Other") |>
  drop_na() |>
  summarize(grp_totpct = sum(pct))
```

```
## # A tibble: 4 × 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:

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

```
## # A tibble: 238 × 21
##   country year      donors  pop pop_d...1  gdp gdp_lag health healt...2 pubhe...3
##   <chr>   <date>      <dbl> <int>  <dbl> <int>  <int>  <dbl>  <dbl>  <dbl>
## 1 Austral... NA          NA   17065  0.220 16774  16591  1300   1224    4.8
## 2 Austral... 1991-01-01  12.1  17284  0.223 17171  16774  1379   1300    5.4
## 3 Austral... 1992-01-01  12.4  17495  0.226 17914  17171  1455   1379    5.4
## 4 Austral... 1993-01-01  12.5  17667  0.228 18883  17914  1540   1455    5.4
## 5 Austral... 1994-01-01  10.2  17855  0.231 19849  18883  1626   1540    5.4
## 6 Austral... 1995-01-01  10.2  18072  0.233 21079  19849  1737   1626    5.5
## 7 Austral... 1996-01-01  10.6  18311  0.237 21923  21079  1846   1737    5.6
## 8 Austral... 1997-01-01  10.3  18518  0.239 22961  21923  1948   1846    5.7
## 9 Austral... 1998-01-01  10.5  18711  0.242 24148  22961  2077   1948    5.9
## 10 Austral... 1999-01-01   8.67 18926  0.244 25445  24148  2231   2077    6.1
## # ... with 228 more rows, 11 more variables: 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>,
## #   and abbreviated variable names 1pop_dens, 2health_lag, 3pubhealth
```

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

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

```
## # A tibble: 30 × 21
##   country year      donors  pop pop_dens  gdp gdp_lag health healt...1 pubhe...2
##   <chr>   <date>      <dbl> <int>    <dbl> <int>  <int>  <dbl>    <dbl>    <dbl>
## 1 Canada 2000-01-01    15.3 30770    0.309 28472 26658 2541    2400    6.3
## 2 Denmark 1992-01-01    16.1  5171   12.0  19644 19126 1660    1603     7
## 3 Ireland 1991-01-01     19   3534    5.03 13495 12917  884    791    4.8
## 4 Ireland 1992-01-01    19.5  3558    5.06 14241 13495 1005    884     5
## 5 Ireland 1993-01-01    17.1  3576    5.09 14927 14241 1041    1005    5.1
## 6 Ireland 1994-01-01    20.3  3590    5.11 15990 14927 1119    1041     5
## 7 Ireland 1995-01-01    24.6  3609    5.14 17789 15990 1208    1119    4.9
## 8 Ireland 1996-01-01    16.8  3636    5.17 19245 17789 1269    1208    4.7
## 9 Ireland 1997-01-01    20.9  3673    5.23 22017 19245 1417    1269    4.8
## 10 Ireland 1998-01-01    23.8  3715    5.29 23995 22017 1487    1417    4.7
## # ... with 20 more rows, 11 more variables: 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>,
## #   and abbreviated variable names 1health_lag, 2pubhealth
```

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

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

```
## # A tibble: 238 × 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 × 8
##   country    year      world  opt consent_law consent_practice consi...1 ccode
##   <chr>      <date>    <chr>  <chr> <chr>      <chr>      <chr>      <chr>
## 1 Australia NA        Liberal In      Informed   Informed   Yes      0z
## 2 Australia 1991-01-01 Liberal In      Informed   Informed   Yes      0z
## 3 Australia 1992-01-01 Liberal In      Informed   Informed   Yes      0z
## 4 Australia 1993-01-01 Liberal In      Informed   Informed   Yes      0z
## 5 Australia 1994-01-01 Liberal In      Informed   Informed   Yes      0z
## 6 Australia 1995-01-01 Liberal In      Informed   Informed   Yes      0z
## 7 Australia 1996-01-01 Liberal In      Informed   Informed   Yes      0z
## 8 Australia 1997-01-01 Liberal In      Informed   Informed   Yes      0z
## 9 Australia 1998-01-01 Liberal In      Informed   Informed   Yes      0z
## 10 Australia 1999-01-01 Liberal In      Informed   Informed   Yes      0z
## # ... with 228 more rows, and abbreviated variable name 1consistent
```

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 × 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()`

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

```
## # A tibble: 28 × 21
##   country year      donors  pop pop_d...1  gdp gdp_lag health healt...2 pubhe...3
##   <chr>   <date>      <dbl> <int> <dbl> <int> <int> <dbl> <dbl> <dbl>
## 1 Austral... NA          NA   17065  0.220 16774  16591  1300  1224  4.8
## 2 Austral... 1991-01-01  12.1  17284  0.223 17171  16774  1379  1300  5.4
## 3 Austral... 1992-01-01  12.4  17495  0.226 17914  17171  1455  1379  5.4
## 4 Austral... 1993-01-01  12.5  17667  0.228 18883  17914  1540  1455  5.4
## 5 Austral... 1994-01-01  10.2  17855  0.231 19849  18883  1626  1540  5.4
## 6 Austral... 1995-01-01  10.2  18072  0.233 21079  19849  1737  1626  5.5
## 7 Austral... 1996-01-01  10.6  18311  0.237 21923  21079  1846  1737  5.6
## 8 Austral... 1997-01-01  10.3  18518  0.239 22961  21923  1948  1846  5.7
## 9 Austral... 1998-01-01  10.5  18711  0.242 24148  22961  2077  1948  5.9
## 10 Austral... 1999-01-01  8.67  18926  0.244 25445  24148  2231  2077  6.1
## # ... with 18 more rows, 11 more variables: 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>,
## # and abbreviated variable names 1pop_dens, 2health_lag, 3pubhealth
```

This could get cumbersome fast.

Use `%in%` for multiple selections

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

```
organdata |>
```

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

```
## # A tibble: 56 × 21
```

```
##   country year      donors  pop pop_d...1  gdp gdp_lag health healt...2 pubhe...3
##   <chr>   <date>      <dbl> <int>  <dbl> <int>  <int>  <dbl>  <dbl>  <dbl>
## 1 Austral... NA          NA   17065  0.220 16774  16591  1300   1224    4.8
## 2 Austral... 1991-01-01  12.1  17284  0.223 17171  16774  1379   1300    5.4
## 3 Austral... 1992-01-01  12.4  17495  0.226 17914  17171  1455   1379    5.4
## 4 Austral... 1993-01-01  12.5  17667  0.228 18883  17914  1540   1455    5.4
## 5 Austral... 1994-01-01  10.2  17855  0.231 19849  18883  1626   1540    5.4
## 6 Austral... 1995-01-01  10.2  18072  0.233 21079  19849  1737   1626    5.5
## 7 Austral... 1996-01-01  10.6  18311  0.237 21923  21079  1846   1737    5.6
## 8 Austral... 1997-01-01  10.3  18518  0.239 22961  21923  1948   1846    5.7
## 9 Austral... 1998-01-01  10.5  18711  0.242 24148  22961  2077   1948    5.9
## 10 Austral... 1999-01-01   8.67 18926  0.244 25445  24148  2231   2077    6.1
```

```
## # ... with 46 more rows, 11 more variables: 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>,
## #   and abbreviated variable names 1pop_dens, 2health_lag, 3pubhealth
```


Negating %in%

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

```
organdata |>
```

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

```
## # A tibble: 182 × 21
```

```
##   country year      donors  pop pop_dens  gdp gdp_lag health healt...1 pubhe...2
##   <chr>   <date>    <dbl> <int>    <dbl> <int>  <int>  <dbl>   <dbl>   <dbl>
## 1 Austria NA         NA    7678    9.16 18914   17425   1344    1255    5.2
## 2 Austria 1991-01-01  27.6  7755    9.25 19860   18914   1419    1344    5.2
## 3 Austria 1992-01-01  23.1  7841    9.35 20601   19860   1551    1419    5.5
## 4 Austria 1993-01-01  26.2  7906    9.43 21119   20601   1674    1551    5.9
## 5 Austria 1994-01-01  21.4  7936    9.46 21940   21119   1739    1674    5.9
## 6 Austria 1995-01-01  21.5  7948    9.48 22817   21940   1865    1739    5.8
## 7 Austria 1996-01-01  24.7  7959    9.49 23798   22817   1986    1865    5.8
## 8 Austria 1997-01-01  19.5  7968    9.50 24364   23798   1848    1986    5.3
## 9 Austria 1998-01-01  20.7  7977    9.51 25423   24364   1953    1848    5.4
## 10 Austria 1999-01-01  25.9  7992    9.53 26513   25423   2069    1953    5.4
```

```
## # ... with 172 more rows, 11 more variables: 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>,
## #   and abbreviated variable names 1health_lag, 2pubhealth
```

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 × 21  
##   country year      donors  pop pop_dens  gdp gdp_lag health healt...1 pubhe...2  
##   <chr>   <date>    <dbl> <int>    <dbl> <int>  <int>  <dbl>    <dbl>    <dbl>  
## 1 Austria NA         NA    7678    9.16 18914   17425   1344    1255    5.2  
## 2 Austria 1991-01-01  27.6  7755    9.25 19860   18914   1419    1344    5.2  
## 3 Austria 1992-01-01  23.1  7841    9.35 20601   19860   1551    1419    5.5  
## 4 Austria 1993-01-01  26.2  7906    9.43 21119   20601   1674    1551    5.9  
## 5 Austria 1994-01-01  21.4  7936    9.46 21940   21119   1739    1674    5.9  
## 6 Austria 1995-01-01  21.5  7948    9.48 22817   21940   1865    1739    5.8  
## 7 Austria 1996-01-01  24.7  7959    9.49 23798   22817   1986    1865    5.8  
## 8 Austria 1997-01-01  19.5  7968    9.50 24364   23798   1848    1986    5.3  
## 9 Austria 1998-01-01  20.7  7977    9.51 25423   24364   1953    1848    5.4  
## 10 Austria 1999-01-01  25.9  7992    9.53 26513   25423   2069    1953    5.4  
## # ... with 172 more rows, 11 more variables: 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>,  
## #   and abbreviated variable names 1health_lag, 2pubhealth
```

Do more than one thing

Earlier we saw this:

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

A tibble: 34 × 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

Do 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 × 7
## # Groups:   consent_law [2]
##   consent_law country      donors_mean donors_sd gdp_mean health_mean roads...1
##   <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 Kingdom 13.5       0.775   21359.     1561.     67.9
## 8 Informed    United States  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
```

Do 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 × 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.
```

Let's look at that again

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

Let's look at that again

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

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

```
## # A tibble: 238 × 21  
##   country year      donors pop pop_d...1 gdp gdp_lag health healt...2 pub  
##   <chr>   <date>    <dbl> <int>  <dbl> <int>   <int> <dbl>   <dbl> <dbl> <dbl>  
## 1 Austral... NA          NA    17065  0.220 16774   16591  1300    1224  
## 2 Austral... 1991-01-01  12.1  17284  0.223 17171   16774  1379    1300  
## 3 Austral... 1992-01-01  12.4  17495  0.226 17914   17171  1455    1379  
## 4 Austral... 1993-01-01  12.5  17667  0.228 18883   17914  1540    1455  
## 5 Austral... 1994-01-01  10.2  17855  0.231 19849   18883  1626    1540  
## 6 Austral... 1995-01-01  10.2  18072  0.233 21079   19849  1737    1626  
## 7 Austral... 1996-01-01  10.6  18311  0.237 21923   21079  1846    1737  
## 8 Austral... 1997-01-01  10.3  18518  0.239 22961   21923  1948    1846  
## 9 Austral... 1998-01-01  10.5  18711  0.242 24148   22961  2077    1948  
## 10 Austral... 1999-01-01   8.67 18926  0.244 25445   24148  2231    2077  
## # ... with 228 more rows, 11 more variables: 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>,  
## #   and abbreviated variable names 1pop_dens, 2health_lag, 3pubhealth
```


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)
```

```
## # A tibble: 238 × 21  
## # Groups:   consent_law, country [17]  
##   country year      donors  pop pop_d...1  gdp gdp_lag health healt...2 pub...  
##   <chr> <date>      <dbl> <int> <dbl> <int> <int> <dbl> <dbl> <dbl>  
## 1 Austral... NA          NA    17065  0.220 16774  16591  1300  1224  
## 2 Austral... 1991-01-01  12.1  17284  0.223 17171  16774  1379  1300  
## 3 Austral... 1992-01-01  12.4  17495  0.226 17914  17171  1455  1379  
## 4 Austral... 1993-01-01  12.5  17667  0.228 18883  17914  1540  1455  
## 5 Austral... 1994-01-01  10.2  17855  0.231 19849  18883  1626  1540  
## 6 Austral... 1995-01-01  10.2  18072  0.233 21079  19849  1737  1626  
## 7 Austral... 1996-01-01  10.6  18311  0.237 21923  21079  1846  1737  
## 8 Austral... 1997-01-01  10.3  18518  0.239 22961  21923  1948  1846  
## 9 Austral... 1998-01-01  10.5  18711  0.242 24148  22961  2077  1948  
## 10 Austral... 1999-01-01   8.67 18926  0.244 25445  24148  2231  2077  
## # ... with 228 more rows, 11 more variables: 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>,  
## #   and abbreviated variable names 1pop_dens, 2health_lag, 3pubhealth
```

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 × 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 × 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

```
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 × 11
## # Groups:   consent_law [2]
##   conse...1 country gdp_avg gdp_sd gdp_md donor...2 donor...3 donor...4 roads...5 roads...6
##   <chr>    <chr>    <dbl> <dbl> <int>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 Inform... Austra... 22179. 1.57e7 21923    10.6     1.31     10.4    105.     205.
## 2 Inform... Canada  23711. 1.57e7 22764    14.0     0.564    14.0    109.     313.
## 3 Inform... Denmark 23722. 1.52e7 23548    13.1     2.16     12.9    102.     154.
## 4 Inform... Germany 22163. 6.26e6 22164    13.0     0.374    13      113.     671.
## 5 Inform... Ireland 20824. 4.45e7 19245    19.8     6.14     19.2    118.     116.
## 6 Inform... Nether... 23013. 1.42e7 22541    13.7     2.41     13.8     76.1     98.6
## 7 Inform... United... 21359. 1.54e7 20839    13.5     0.601    13.5     67.9    110.
## 8 Inform... United... 29212. 2.09e7 28772    20.0     1.76     20.1    155.     69.8
## 9 Presum... Austria 23876. 1.12e7 23798    23.5     5.84     23.8    150.     917.
## 10 Presum... Belgium 22500. 1.01e7 22152    21.9     3.75     21.4    155.     423.
## 11 Presum... Finland 21019. 1.35e7 19842    18.4     2.33     19.4     93.6    361.
## 12 Presum... France  22603. 1.06e7 21990    16.8     2.55     16.6    156.     403.
## 13 Presum... Italy   21554. 7.74e6 21396    11.1     18.3     11.3    122.     103.
## 14 Presum... Norway 26448. 4.21e7 26218    15.4     1.23     15.4     70.0     44.6
## 15 Presum... Spain  16933. 8.34e6 16416    28.1     24.6     28      161.    1243.
## 16 Presum... Sweden 22415. 1.03e7 22029    13.1     3.07     12.7     72.3     175.
```

It's OK to use the function names

```
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 × 11
## # Groups:   consent_law [2]
##   consent_law country    gdp_m...1 gdp_var gdp_m...2 donor...3 donor...4 donor...5 roads...6
##   <chr>         <chr>      <dbl>    <dbl>    <int>    <dbl>    <dbl>    <dbl>    <dbl>
## 1 Informed      Australia  22179.  1.57e7  21923    10.6    1.31    10.4    105.
## 2 Informed      Canada    23711.  1.57e7  22764    14.0    0.564   14.0    109.
## 3 Informed      Denmark   23722.  1.52e7  23548    13.1    2.16    12.9    102.
## 4 Informed      Germany   22163.  6.26e6  22164    13.0    0.374    13     113.
## 5 Informed      Ireland   20824.  4.45e7  19245    19.8    6.14    19.2    118.
## 6 Informed      Netherla... 23013.  1.42e7  22541    13.7    2.41    13.8    76.1
## 7 Informed      United K... 21359.  1.54e7  20839    13.5    0.601    13.5    67.9
## 8 Informed      United S... 29212.  2.09e7  28772    20.0    1.76    20.1    155.
## 9 Presumed      Austria    23876.  1.12e7  23798    23.5    5.84    23.8    150.
## 10 Presumed     Belgium    22500.  1.01e7  22152    21.9    3.75    21.4    155.
## 11 Presumed     Finland    21019.  1.35e7  19842    18.4    2.33    19.4    93.6
## 12 Presumed     France     22603.  1.06e7  21990    16.8    2.55    16.6    156.
## 13 Presumed     Italy      21554.  7.74e6  21396    11.1    18.3    11.3    122.
## 14 Presumed     Norway     26448.  4.21e7  26218    15.4    1.23    15.4    70.0
## 15 Presumed     Spain      16933.  8.34e6  16416    28.1    24.6    28     161.
## 16 Presumed     Sweden     22415.  1.03e7  22029    13.1    3.07    12.7    72.3
```

Selection with `across(where())`

```
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 × 41
## # Groups:   consent_law [2]
##   consent_law country  donors...1 donor...2 donor...3 pop_m...4 pop_var pop_m...5 pop_d...6
##   <chr>         <chr>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <int>    <dbl>
## 1 Informed     Australia    10.6    1.31    10.4   18318.   6.90e5   18311    0.237
## 2 Informed     Canada      14.0    0.564   14.0   29608.   1.42e6   29672    0.297
## 3 Informed     Denmark     13.1    2.16    12.9    5257.   6.50e3    5263    12.2
## # ... with 14 more rows, 32 more variables: 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>, ...
```

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)
```

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".

```
## # A tibble: 17 × 41
## # Groups:   consent_law [2]
##   consent_law country  mean_d...1 var_d...2 media...3 mean_...4 var_pop media...5 mean_...6
##   <chr>         <chr>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <int>    <dbl>
## 1 Informed     Australia    10.6     1.31     10.4   18318.    6.90e5    18311    0.237
## 2 Informed     Canada      14.0     0.564     14.0   29608.    1.42e6    29672    0.297
## 3 Informed     Denmark     13.1     2.16     12.9    5257.    6.50e3     5263    12.2
## # ... with 14 more rows, 32 more variables: 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>, ...
```

This all works with `mutate()`, too

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

```
## # A tibble: 238 × 7
##   country world opt consent_law consent_practice consistent ccode
##   <chr>    <chr> <chr> <chr>          <chr>          <chr>    <chr>
## 1 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 2 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 3 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 4 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 5 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 6 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 7 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 8 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 9 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## 10 AUSTRALIA LIBERAL IN    INFORMED    INFORMED        YES      0Z
## # ... with 228 more rows
```


Arrange rows and columns

Sort rows with `arrange()`

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

```
## # A tibble: 17 × 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
```

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

```
## # A tibble: 17 × 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 × 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()`.