Coding Lab: Manipulating data with `dplyr`

Ari Anisfeld

Summer 2020

Data manipulation with `dplyr`

Once you have data in R, you'll want to explore it.

The `tidyverse` package `dplyr` provides a toolkit for data manipulation.

We will cover:

- `select()` to pick columns
- `arrange()` to order the data
- `mutate()` to create new columns
- `filter()` to get rows that meet a criteria
- `summarize()` to summarize data

selecting columns with `select()`

select()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

selecting columns with `select()`

Use case: You want to present a subset of your columns

```
select(texas_housing_data, city, date, sales, listings)
```

```
## # A tibble: 8,602 x 4
##    city      date sales listings
##    <chr>    <dbl> <dbl>    <dbl>
##  1 Abilene   2000    72      701
##  2 Abilene  2000.    98      746
##  3 Abilene  2000.   130      784
##  4 Abilene  2000.    98      785
##  5 Abilene  2000.   141      794
##  6 Abilene  2000.   156      780
##  7 Abilene  2000.   152      742
##  8 Abilene  2001.   131      765
##  9 Abilene  2001.   104      771
## 10 Abilene  2001.   101      764
## # ... with 8,592 more rows
```

selecting columns with `select()`

Use case: You want to present a subset of your columns

```
select(texas_housing_data, -c(city, date, sales, listings))
```

The – says to exclude the columns listed in the vector.

selecting columns with `select()`, helpers

Use case: You want to reorder your columns

```
select(texas_housing_data, city, date,
       sales, listings, everything())
```

```
## # A tibble: 8,602 x 9
##    city      date sales listings  year month   volume mec
##    <chr>    <dbl> <dbl>    <dbl> <dbl> <dbl>    <dbl> <c
##  1 Abilene   2000    72      701  2000     1  5380000  71
##  2 Abilene  2000.    98      746  2000     2  6505000  58
##  3 Abilene  2000.   130      784  2000     3  9285000  58
##  4 Abilene  2000.    98      785  2000     4  9730000  68
##  5 Abilene  2000.   141      794  2000     5 10590000  67
##  6 Abilene  2000.   156      780  2000     6 13910000  66
##  7 Abilene  2000.   152      742  2000     7 12635000  73
##  8 Abilene  2001.   131      765  2000     8 10710000  75
##  9 Abilene  2001.   104      771  2000     9  7615000  64
## 10 Abilene  2001.   101      764  2000    10  7040000  59
## # ... with 8,592 more rows
```

sort rows with `arrange()`

arrange()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

→

storm	wind	pressure	date
Ana	40	1013	1997-07-01
Alex	45	1009	1998-07-30
Arthur	45	1010	1996-06-21
Arlene	50	1010	1999-06-13
Allison	65	1005	1995-06-04
Alberto	110	1007	2000-08-12

sort rows with arrange()

```
arrange(texas_housing_data, year)
```

```
## # A tibble: 8,602 x 9
##    city      year month sales    volume median listings in
##    <chr>    <dbl> <dbl> <dbl>     <dbl>  <dbl>    <dbl>
##  1 Abilene   2000     1    72   5380000  71400      701
##  2 Abilene   2000     2    98   6505000  58700      746
##  3 Abilene   2000     3   130   9285000  58100      784
##  4 Abilene   2000     4    98   9730000  68600      785
##  5 Abilene   2000     5   141  10590000  67300      794
##  6 Abilene   2000     6   156  13910000  66900      780
##  7 Abilene   2000     7   152  12635000  73500      742
##  8 Abilene   2000     8   131  10710000  75000      765
##  9 Abilene   2000     9   104   7615000  64500      771
## 10 Abilene   2000    10   101   7040000  59300      764
## # ... with 8,592 more rows
```

sort rows with arrange()

To change the order of use desc()

```r
arrange(texas_housing_data, desc(year))
```

```
## # A tibble: 8,602 x 9
##    city      year month sales   volume median listings
##    <chr>    <dbl> <dbl> <dbl>    <dbl>  <dbl>    <dbl>
##  1 Abilene   2015     1   158 23486998 134100      801
##  2 Abilene   2015     2   151 19834263 126500      767
##  3 Abilene   2015     3   198 31869437 136800      821
##  4 Abilene   2015     4   201 28301159 129600      891
##  5 Abilene   2015     5   199 31385757 144700      919
##  6 Abilene   2015     6   260 41396230 141500      965
##  7 Abilene   2015     7   268 45845730 148700      986
##  8 Amarillo  2015     1   204 33188726 138500     1120
##  9 Amarillo  2015     2   188 34355428 149400     1084
## 10 Amarillo  2015     3   317 53603130 140900     1051
## # ... with 8,592 more rows
```

Ceci n'est pas une pipe.

Interlude: Ceci est une %>%

The pipe %>% operator takes the left-hand side and makes it *input* in the right-hand side.

- ▶ by default, the left-hand side is the *first argument* of the right-hand side function.

```r
# a tibble is the first argument
select(texas_housing_data, city, year, sales, volume)

texas_housing_data %>%
  select(city, year, sales, volume)
```

Ceci est une %>%

We can chain together tidyverse functions to avoid making so many intermediate data frames!

```
texas_housing_data %>%
  select(city, year, month, median) %>%
  arrange(desc(median))
```

```
## # A tibble: 8,602 x 4
##    city          year month median
##    <chr>        <dbl> <dbl>  <dbl>
##  1 Collin County 2015     5 304200
##  2 Collin County 2015     6 300400
##  3 Collin County 2015     7 292600
##  4 Collin County 2015     4 291400
##  5 Collin County 2015     3 285800
##  6 Fort Bend     2015     6 284200
##  7 Collin County 2015     2 283400
##  8 Midland       2014     6 283100
##  9 Fort Bend     2014     6 282300
```

creating columns with `mutate()`

mutate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date	ratio	inverse
Alberto	110	1007	2000-08-12	9.15	0.11
Alex	45	1009	1998-07-30	22.42	0.04
Allison	65	1005	1995-06-04	15.46	0.06
Ana	40	1013	1997-07-01	25.32	0.04
Arlene	50	1010	1999-06-13	20.20	0.05
Arthur	45	1010	1996-06-21	22.44	0.04

creating columns with `mutate()`

```
texas_housing_data %>%
  mutate(mean_price = volume / sales) %>%
  select(city, year, month, mean_price, sales, volume)

## # A tibble: 8,602 x 6
##    city     year month mean_price sales   volume
##    <chr>   <dbl> <dbl>      <dbl> <dbl>    <dbl>
##  1 Abilene  2000     1     74722.    72  5380000
##  2 Abilene  2000     2     66378.    98  6505000
##  3 Abilene  2000     3     71423.   130  9285000
##  4 Abilene  2000     4     99286.    98  9730000
##  5 Abilene  2000     5     75106.   141 10590000
##  6 Abilene  2000     6     89167.   156 13910000
##  7 Abilene  2000     7     83125    152 12635000
##  8 Abilene  2000     8     81756.   131 10710000
##  9 Abilene  2000     9     73221.   104  7615000
## 10 Abilene  2000    10     69703.   101  7040000
## # ... with 8,592 more rows
```

Binary operators: Math in R

R is a calculator! We can do math with numbers, using the following symbols:

```
4 + 4
4 - 4
4 * 4
4 / 4
4 ^ 4
5 %% 4   # gives the remainder after dividing
```

creating columns with `mutate()`

When we mutate, you can create new columns.

- ▶ On the right side of the equal sign, you have the name of a new column.
- ▶ On the left side, you have code that creates a new column (using vector operations)[1]

```
texas_housing_data %>%
  mutate(mean_price = volume / sales) %>%
  select(city, year, month,  mean_price, sales, volume)
```

```
## # A tibble: 8,602 x 6
##    city    year month mean_price sales   volume
##    <chr>  <dbl> <dbl>      <dbl> <dbl>    <dbl>
## 1 Abilene 2000     1     74722.    72  5380000
## 2 Abilene 2000     2     66378.    98  6505000
## 3 Abilene 2000     3     71423.   130  9285000
## 4 Abilene 2000     4     99286.    98  9730000
## 5 Abilene 2000     5     75106.   141 10590000
## 6 Abilene 2000     6     89167.   156 13910000
```

creating columns with `mutate()`

You can create multiple columns at a single time and even use information from a newly created column as input.

```
texas_housing_data %>%
  mutate(mean_price = volume / sales,
         sqrt_mean_price = sqrt(mean_price)) %>%
  select(city, year, month,  mean_price, sales, volume)
```

```
## # A tibble: 8,602 x 6
##    city     year month mean_price sales   volume
##    <chr>   <dbl> <dbl>      <dbl> <dbl>    <dbl>
##  1 Abilene  2000     1     74722.    72  5380000
##  2 Abilene  2000     2     66378.    98  6505000
##  3 Abilene  2000     3     71423.   130  9285000
##  4 Abilene  2000     4     99286.    98  9730000
##  5 Abilene  2000     5     75106.   141 10590000
##  6 Abilene  2000     6     89167.   156 13910000
##  7 Abilene  2000     7     83125    152 12635000
##  8 Abilene  2000     8     81756.   131 10710000
```

choose rows that match a condition with `filter()`

filter()

storms

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Ana	40	1013	1997-07-01

choose rows that match a condition with `filter()`

Get all the data from 2013

```
filter(texas_housing_data, year == 2013)
```

```
## # A tibble: 552 x 9
##    city     year month sales   volume median listings in
##    <chr>   <dbl> <dbl> <dbl>    <dbl>  <dbl>    <dbl>
##  1 Abilene  2013     1   114 15794494 125300      966
##  2 Abilene  2013     2   140 16552641  94400      943
##  3 Abilene  2013     3   164 19609711 102500      958
##  4 Abilene  2013     4   213 27261796 113700      948
##  5 Abilene  2013     5   225 31901380 130000      923
##  6 Abilene  2013     6   209 29454125 127300      960
##  7 Abilene  2013     7   218 32547446 140000      969
##  8 Abilene  2013     8   236 30777727 120000      976
##  9 Abilene  2013     9   195 26237106 127500      985
## 10 Abilene  2013    10   167 21781187 119000      993
## # ... with 542 more rows
```

Relational operators return TRUE or FALSE

Before moving forward with `filter()`, we need to know about relational operators and logical operators

Operator	Name
$<$	less than
$>$	greater than
$<=$	less than or equal to
$>=$	greater than or equal to
$==$	equal to
$!=$	not equal to
%in%	matches something in

Relational operators in practice

```
4 < 4
```

```
## [1] FALSE
```

```
4 >= 4
```

```
## [1] TRUE
```

```
4 == 4
```

```
## [1] TRUE
```

```
4 != 4
```

```
## [1] FALSE
```

```
4 %in% c(1, 2, 3)
```

```
## [1] FALSE
```

logical operators combine TRUEs and FALSEs logically

Operator	Name
!	not
&	and
|	or

```
# not true
! TRUE
```

```
## [1] FALSE
```

```
# are both x & y TRUE?
TRUE  &  FALSE
```

```
## [1] FALSE
```

```
# is either x | y TRUE?
TRUE | FALSE
```

```
## [1] TRUE
```

What do the following return?

Logical operators team up with relational operators.

- ▶ First, evaluate the relational operator
- ▶ Then, care out the logic.

```
! (4 > 3) # ! TRUE
(5 > 1) & (5 > 2) # TRUE & TRUE
(4 > 10) | (20 > 3) # FALSE | TRUE
```

This is hard to wrap your head around. We'll have plenty of practice!

choose rows that match a condition with `filter()`

Get all the data from 2013 for Houston.

- ▶ in `filter()` additional match criteria are treated like `and`

```
texas_housing_data %>%
  filter(year == 2013,
         city == "Houston")
```

```
## # A tibble: 12 x 9
##    city      year month sales      volume median listings
##    <chr>    <dbl> <dbl> <dbl>       <dbl>  <dbl>    <dbl>
##  1 Houston   2013     1  4273   852045057 149500    21364
##  2 Houston   2013     2  4886  1060985674 161900    21293
##  3 Houston   2013     3  6382  1479273481 172300    20909
##  4 Houston   2013     4  7116  1770746764 182400    20607
##  5 Houston   2013     5  8439  2121508529 186100    20526
##  6 Houston   2013     6  7935  2073909387 191600    21008
##  7 Houston   2013     7  8468  2168720825 187800    21497
##  8 Houston   2013     8  8155  2083377894 186700    21366
## 9 Houston   2013     9  6706  1638993780 180900    21807
```

choose rows that match a condition with `filter()`

Get all the data from 2013 for Houston or Austin

- in `filter()` additional match criteria are treated like and
- we get nothing returned here, because no observation is in Houston AND in Austin.

```
texas_housing_data %>%
  filter(year == 2013,
         city == "Houston", city == "Austin")
```

```
## # A tibble: 0 x 9
## # ... with 9 variables: city <chr>, year <dbl>, month <d
## #   volume <dbl>, median <dbl>, listings <dbl>, invento
```

choose rows that match a condition with `filter()`

Get all the data from after than 2013 for Houston OR Austin

```
texas_housing_data %>%
  filter(year > 2013,
         city == "Houston" | city == "Austin")
```

```
## # A tibble: 38 x 9
##    city    year month sales      volume median listings i
##    <chr>  <dbl> <dbl> <dbl>       <dbl>  <dbl>    <dbl>
##  1 Austin  2014     1  1582   426127544 213700     5118
##  2 Austin  2014     2  1903   550882376 229400     5255
##  3 Austin  2014     3  2434   717821612 235600     5512
##  4 Austin  2014     4  2691   813253968 237000     5838
##  5 Austin  2014     5  3178  1012123948 243900     6539
##  6 Austin  2014     6  3195  1023051880 248900     7040
##  7 Austin  2014     7  3151   982086356 246900     7475
##  8 Austin  2014     8  3023   927019222 243800     7326
##  9 Austin  2014     9  2664   813797562 238900     7072
## 10 Austin  2014    10  2588   796863816 239600     6791
```

choose rows that match a condition with `filter()`

Get all the data from after than 2013 for Houston Galveston

```
texas_housing_data %>%
  filter(year > 2013,
         city %in% c("Houston", "Dallas", "Austin"))
```

```
## # A tibble: 57 x 9
##    city    year month sales     volume median listings
##    <chr>  <dbl> <dbl> <dbl>      <dbl>  <dbl>    <dbl>
##  1 Austin  2014     1  1582  426127544 213700     5118
##  2 Austin  2014     2  1903  550882376 229400     5255
##  3 Austin  2014     3  2434  717821612 235600     5512
##  4 Austin  2014     4  2691  813253968 237000     5838
##  5 Austin  2014     5  3178 1012123948 243900     6539
##  6 Austin  2014     6  3195 1023051880 248900     7040
##  7 Austin  2014     7  3151  982086356 246900     7475
##  8 Austin  2014     8  3023  927019222 243800     7326
##  9 Austin  2014     9  2664  813797562 238900     7072
## 10 Austin  2014    10  2588  796863816 239600     6791
```

summarize data with `summarize()`

city	particle size	amount (μg/m³)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

median
22.5

summarize data with summarize()

Calculate total volume of sales in Texas from 2014.

```
texas_housing_data %>%
  filter(year == 2014) %>%
  summarize(total_volume = sum(volume))
```

```
## # A tibble: 1 x 1
##    total_volume
##           <dbl>
## 1   84760948831
```

summarize data with `summarize()`

Calculate the mean and median number of sales in Texas's three largest cities.

```
texas_housing_data %>%
  filter(city %in%
         c("Houston", "Dallas", "San Antonio")) %>%
  summarize(median_n_sales = median(sales),
            mean_n_sales = mean(sales))
```

```
## # A tibble: 1 x 2
##   median_n_sales mean_n_sales
##            <dbl>        <dbl>
## 1           3996        3890.
```

summarize data with `summarize()`

There are many useful functions that go with summarize. Try `?summarize` for more.

```
texas_housing_data %>%
  filter(city %in%
         c("Houston", "Dallas", "San Antonio")) %>%
  summarize(n_obs = n(),
            n_cities = n_distinct(city))
```

```
## # A tibble: 1 x 2
##   n_obs n_cities
##   <int>    <int>
## 1   561        3
```

summarize data with `summarize()`

If you try to make a summarize statistic that does not collapse the data to a single value (per group), you'll get an error like so:

```
texas_housing_data %>%
  filter(city %in%
          c("Houston", "Dallas", "San Antonio")) %>%
  summarize(mean_price = volume / sales)

Error: Column `mean_price` must be length 1 (a summary valu
```

Get number of observations

Recap: manipulating data with `dplyr`

We learned

- how to employ the 5 `dplyr` verbs of highest importance including
  - `select()` to pick columns
  - `arrange()` to order the data
  - `mutate()` to create new columns
  - `filter()` to get rows that meet a criteria
  - `summarize()` to summarize data
- how to use relation operators, binary operators for math and logical operators in `dplyr` contexts