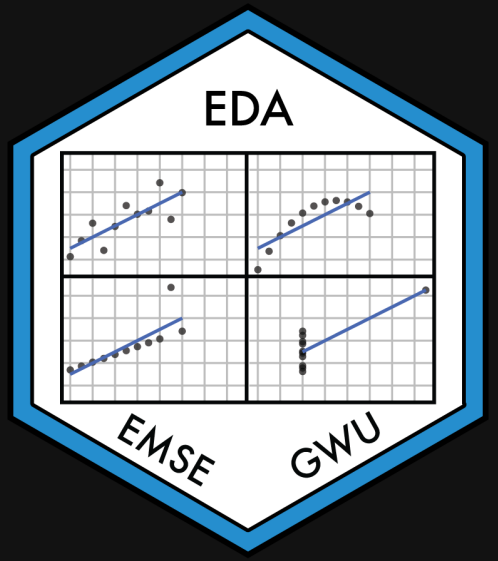


Week 9: *Cleaning Data*

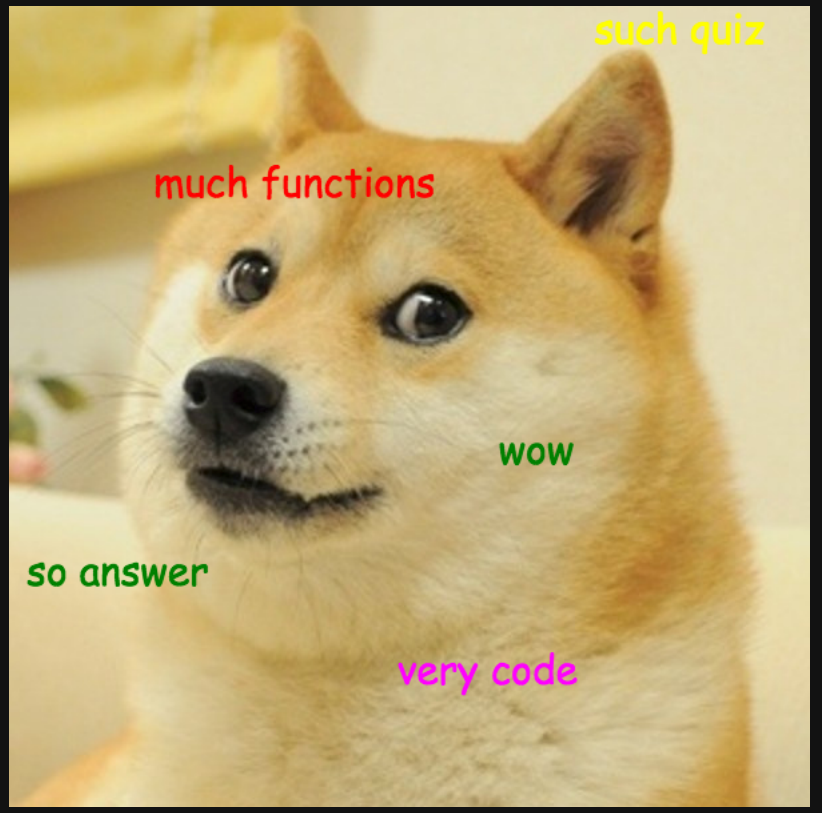
EMSE 4575: Exploratory Data Analysis

John Paul Helveston

March 10, 2021

Quiz 3

Link is on the schedule



10:00

Waffles and Animations

Waffles

The bug:

```
install.packages("waffle")
```

The fix:

```
install.packages("waffle",
  repos = "https://cinc.rud.is")
```

Animation

The bug:

```
animate(anim_plot)
```

The fix:

```
install.packages('magick')

animate(anim_plot,
  renderer = magick_renderer())
```

Tip of the week

Copy-paste magic with `datapasta`

Useful for "small data": e.g., U.S. State Abbreviations

Today's data

"Clean" data

```
wildlife_impacts <- read_csv(here::here('data', 'wildlife_impacts.csv'))
milk_production  <- read_csv(here::here('data', 'milk_production.csv'))
msleep           <- read_csv(here::here('data', 'msleep.csv'))
```

"Messy" data

```
wind     <- read_excel(here::here('data', 'US_State_Wind_Energy_Facts_2018.xlsx'))
hot_dogs <- read_excel(here::here('data', 'hot_dog_winners.xlsx'))
```

Plus two new packages:

```
# For manipulating dates
install.packages('lubridate')

# For cleaning column names
install.packages('janitor')
```

Week 9: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

BREAK

4. Re-coding variables

5. Dates

Week 9: *Cleaning Data*

1. Merging datasets with joins

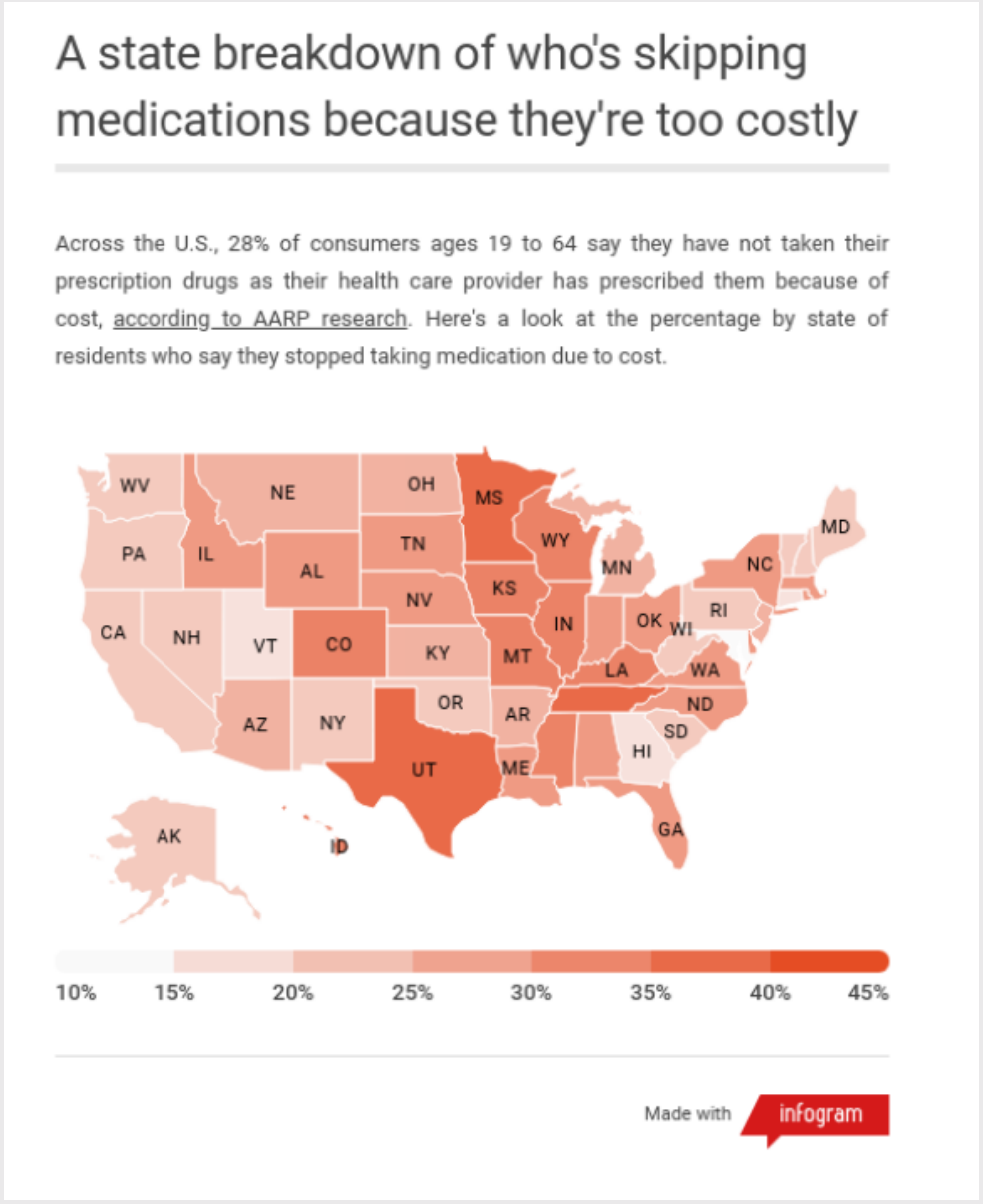
2. Are your variables the right *type*?

3. Are your variables the right *name*?

BREAK

4. Re-coding variables

5. Dates



What's wrong with this map?

Likely culprit: Merging two columns

```
head(names)
```

```
#>   state_name
#> 1    Alabama
#> 2     Alaska
#> 3    Arizona
#> 4   Arkansas
#> 5 California
#> 6   Colorado
```

```
head(abbs)
```

```
#>   state_abb
#> 1        AK
#> 2        AL
#> 3        AR
#> 4        AZ
#> 5        CA
#> 6        CO
```

```
result <- bind_cols(names, abbs)
head(result)
```

```
#>   state_name state_abb
#> 1    Alabama        AK
#> 2     Alaska        AL
#> 3    Arizona        AR
#> 4   Arkansas        AZ
#> 5 California        CA
#> 6   Colorado        CO
```

Joins

1. `inner_join()`
2. `left_join()` / `right_join()`
3. `full_join()`

Example: `band_members` & `band_instruments`

`band_members`

```
#> # A tibble: 3 x 2
#>   name  band
#>   <chr> <chr>
#> 1 Mick  Stones
#> 2 John  Beatles
#> 3 Paul  Beatles
```

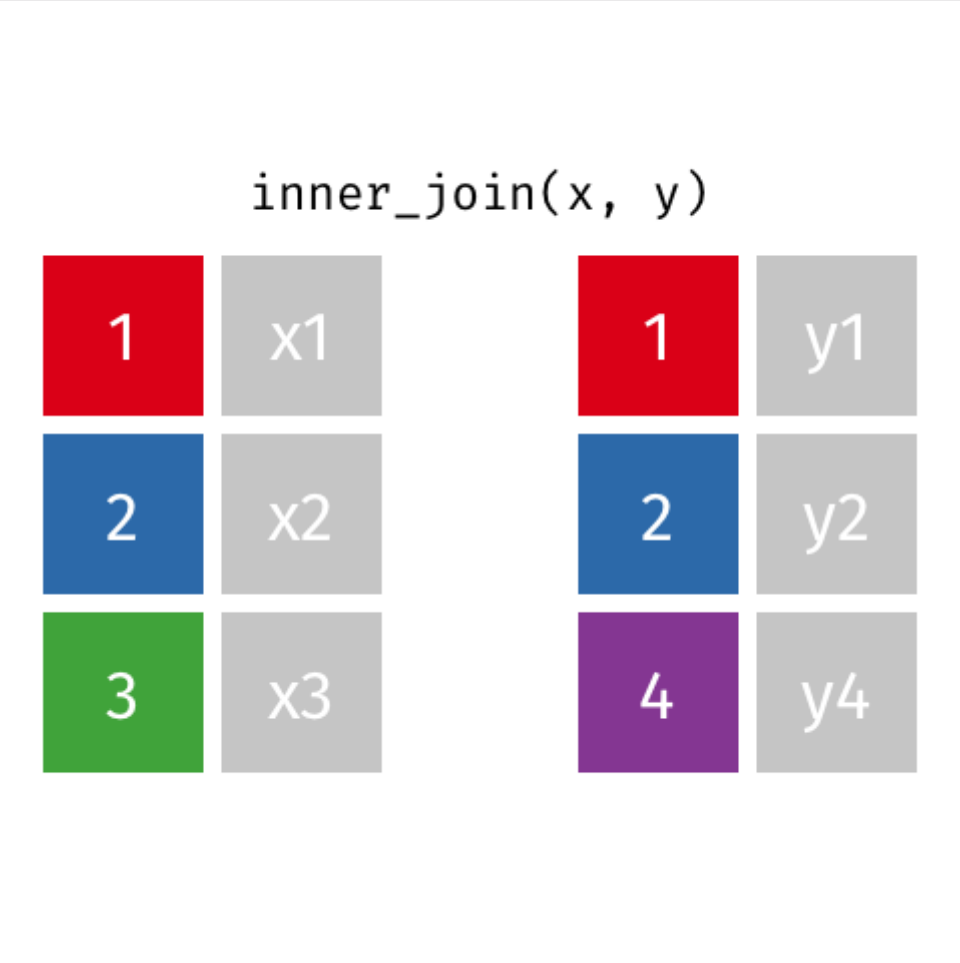
`band_instruments`

```
#> # A tibble: 3 x 2
#>   name  plays
#>   <chr> <chr>
#> 1 John  guitar
#> 2 Paul  bass
#> 3 Keith guitar
```

inner_join()

```
band_members %>%
    inner_join(band_instruments)
```

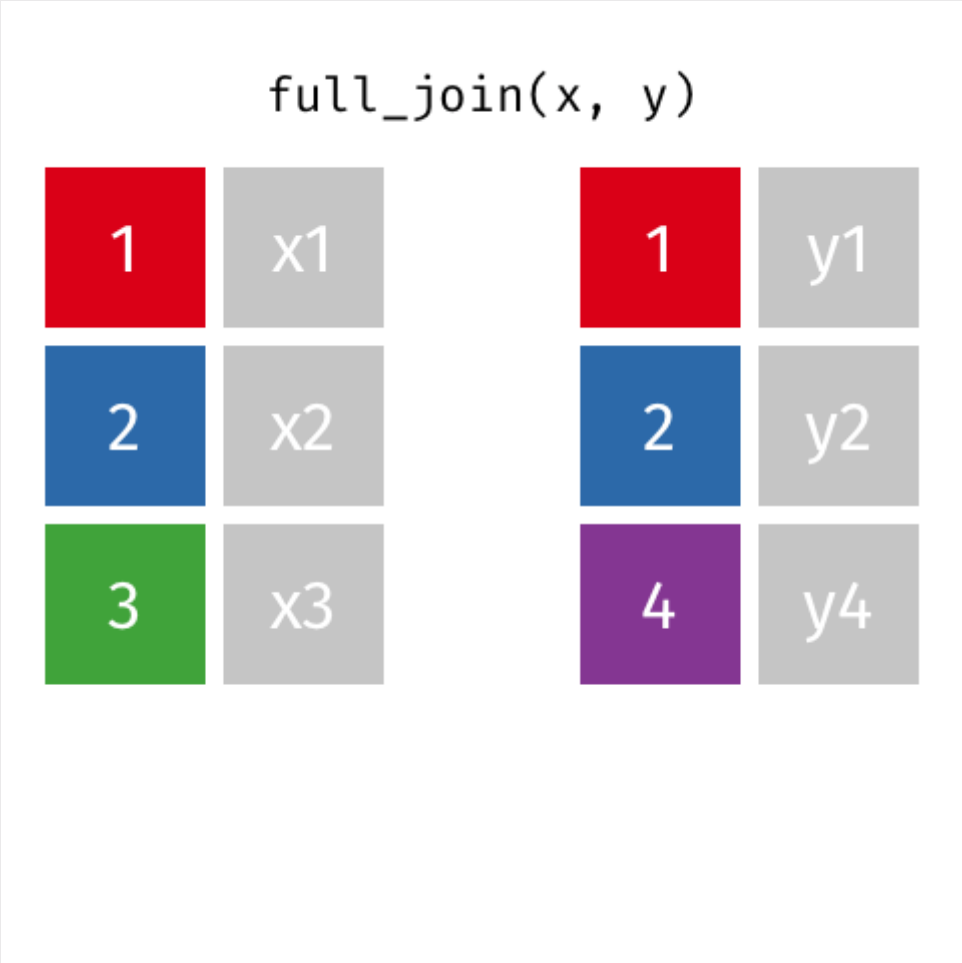
```
#> # A tibble: 2 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 John  Beatles guitar
#> 2 Paul  Beatles bass
```



full_join()

```
band_members %>%
    full_join(band_instruments)
```

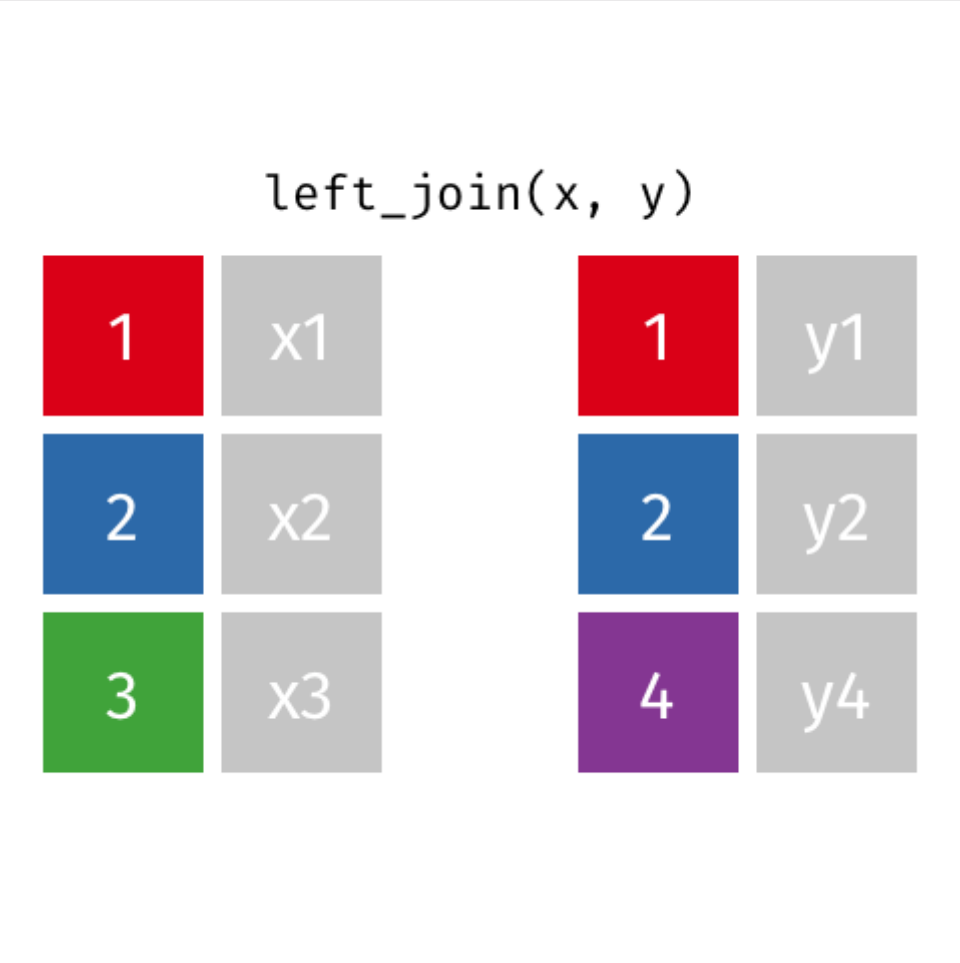
```
#> # A tibble: 4 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
#> 4 Keith <NA>    guitar
```



left_join()

```
band_members %>%
    left_join(band_instruments)
```

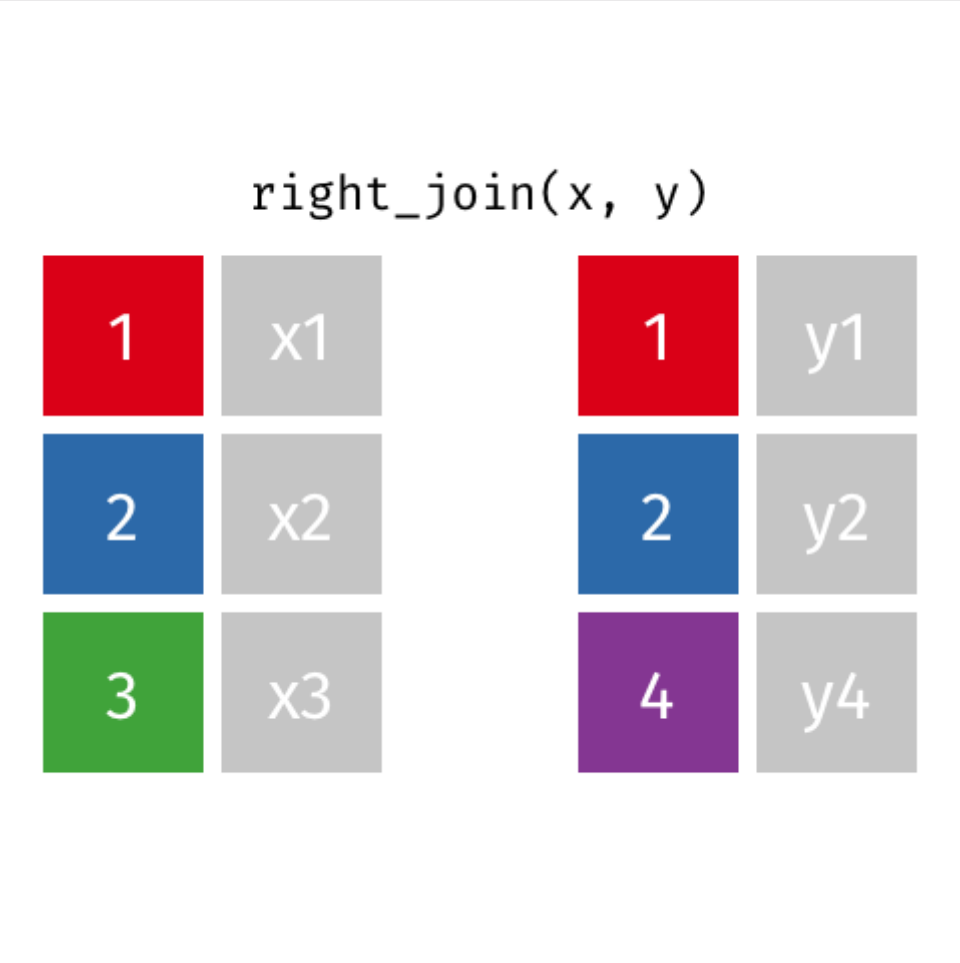
```
#> # A tibble: 3 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```



right_join()

```
band_members %>%
    right_join(band_instruments)
```

```
#> # A tibble: 3 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 John  Beatles guitar
#> 2 Paul  Beatles bass
#> 3 Keith <NA>    guitar
```



Specify the joining variable name

```
band_members %>%
    left_join(band_instruments)
```

```
#> Joining, by = "name"
```

```
#> # A tibble: 3 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```

```
band_members %>%
    left_join(band_instruments,
              by = 'name')
```

```
#> # A tibble: 3 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```

Specify the joining variable name

If the names differ, use `by = c("left_name" = "joining_name")`

```
band_members
```

```
#> # A tibble: 3 x 2
#>   name  band
#>   <chr> <chr>
#> 1 Mick  Stones
#> 2 John  Beatles
#> 3 Paul  Beatles
```

```
band_instruments2
```

```
#> # A tibble: 3 x 2
#>   artist plays
#>   <chr>  <chr>
#> 1 John   guitar
#> 2 Paul   bass
#> 3 Keith  guitar
```

```
band_members %>%
    left_join(band_instruments2,
              by = c("name" = "artist"))
```

```
#> # A tibble: 3 x 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```

Specify the joining variable name

Or just rename the joining variable in a pipe

```
band_members
```

```
#> # A tibble: 3 x 2
#>   name  band
#>   <chr> <chr>
#> 1 Mick  Stones
#> 2 John  Beatles
#> 3 Paul  Beatles
```

```
band_instruments2
```

```
#> # A tibble: 3 x 2
#>   artist plays
#>   <chr>  <chr>
#> 1 John   guitar
#> 2 Paul   bass
#> 3 Keith  guitar
```

```
band_members %>%
    rename(artist = name) %>%
    left_join(band_instruments2,
              by = "artist")
```

```
#> # A tibble: 3 x 3
#>   artist band    plays
#>   <chr>  <chr>   <chr>
#> 1 Mick   Stones  <NA>
#> 2 John   Beatles guitar
#> 3 Paul   Beatles bass
```

Your turn

15:00

1) Create a data frame called `state_data` by joining the data frames `states_abbs` and `milk_production` and then selecting the variables `region`, `state_name`, `state_abb`. **Hint**: Use the `distinct()` function to drop repeated rows.

Your result should look like this:

```
head(state_data)
```

```
#> # A tibble: 6 x 3
#>   region    state_name    state_abb
#>   <chr>     <chr>         <chr>
#> 1 Northeast Maine         ME
#> 2 Northeast New Hampshire NH
#> 3 Northeast Vermont       VT
#> 4 Northeast Massachusetts MA
#> 5 Northeast Rhode Island  RI
#> 6 Northeast Connecticut   CT
```

2) Join the `state_data` data frame to the `wildlife_impacts` data frame, adding the variables `region` and `state_name`

```
glimpse(wildlife_impacts)
```

```
#> Rows: 56,978
#> Columns: 24
#> $ region                <chr> "Northeast", "Northeast", "Northeast", "Northeast"
#> $ state_name            <chr> "Maine", "Maine", "Maine", "Maine", "Maine", "Mai
#> $ state_abb             <chr> "ME", "ME", "ME", "ME", "ME", "ME", "ME", "ME", "
#> $ incident_date         <dttm> 2018-10-23, 2018-10-07, 2018-10-05, 2018-10-05, 2
#> $ airport_id            <chr> "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "
#> $ airport               <chr> "PORTLAND INTL JETPORT (ME)", "PORTLAND INTL JETP
#> $ operator              <chr> "AMERICAN AIRLINES", "AMERICAN AIRLINES", "AMERICA
#> $ atype                 <chr> "A-320", "A-319", "A-319", "EMB-190", "EMB-170", "
#> $ type_eng              <chr> "D", "D", "D", "D", "D", "D", "D", "D", "D", "C",
#> $ species_id            <chr> "UNKBS", "ZX302", "ZS010", "I1102", "K3310", "YH0
#> $ species               <chr> "Unknown bird - small", "Swamp sparrow", "Blackpo
#> $ damage                <chr> "N", NA, "N", "M?", "N", "N", "N", "N", "N", "N",
#> $ num_engs              <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
#> $ incident_month        <dbl> 10, 10, 10, 10, 7, 11, 11, 10, 7, 8, 11, 7, 5, 4,
#> $ incident_year         <dbl> 2018, 2018, 2018, 2018, 2017, 2016, 2016, 2016, 20
#> $ time_of_day           <chr> NA, "Night", "Night", "Day", "Dawn", "Day", "Day",
#> $ time                  <dbl> 1310, 1035, 2200, 1645, 645, 1345, 1346, 1400, 110
#> $ height                <dbl> 15, NA, 1000, 0, 0, 0, 0, NA, NA, 2000, 0, 50, 0,
#> $ speed                 <dbl> 150, NA, 140, 110, NA, NA, NA, NA, NA, 250, 100, N
#> $ phase_of_flt          <chr> "departure", "arrival", "arrival", "arrival", "arr
#> $ sky                   <chr> "Overcast", "Some Cloud", "Some Cloud", "Some Clou
#> $ precip                <chr> "None", "None", "None", "None", "None", "None", NA
#> $ cost_repairs_infl_adj <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
#> $ weekday_name          <ord> Tue, Sun, Fri, Fri, Tue, Mon, Mon, Sat, Sat, Wed,
```

Week 9: *Cleaning Data*

1. Merging datasets with joins
2. Are your variables the right *type*?
3. Are your variables the right *name*?

BREAK

4. Re-coding variables
5. Dates

Using the `col_types` argument

- You can change the column type when reading in data
- Different syntax for `readxl::read_excel()` and `readr::read_csv()`

readxl::read_excel()

`col_types` must be a vector describing each column type

```
wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx'))

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ Ranking                          <chr> "1.0", "2.0",
#> $ State                            <chr> "TEXAS", "OKLA
#> $ `Installed Capacity (MW)`        <dbl> 23262, 7495, 7
#> $ `Equivalent Homes Powered`       <chr> "6235000.0", "
#> $ `Total Investment ($ Millions)` <chr> "42000.0", "13
#> $ `Wind Projects Online`           <dbl> 136, 45, 107,
#> $ `# of Wind Turbines`             <chr> "12750.0", "37
```

readxl::read_excel()

`col_types` must be a vector describing each column type

How it is in Excel	How it will be in R	How to request in `col_types`
anything	*non-existent*	`"skip"`
empty	`logical`, but all `NA`	*you cannot request this*
boolean	`logical`	`"logical"`
numeric	`numeric`	`"numeric"`
datetime	`POSIXct`	`"date"`
text	`character`	`"text"`
anything	`list`	`"list"`

```
columns <- c('numeric', 'text', rep('numeric', 5))
columns
```

```
#> [1] "numeric" "text"    "numeric" "numeric" "numer
```

```
wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx'),
  col_types = columns)

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ Ranking                          <dbl> 1, 2, 3, 4
#> $ State                            <chr> "TEXAS", "
#> $ `Installed Capacity (MW)`        <dbl> 23262, 749
#> $ `Equivalent Homes Powered`       <dbl> 6235000, 2
#> $ `Total Investment ($ Millions)` <dbl> 42000, 137
#> $ `Wind Projects Online`           <dbl> 136, 45, 1
#> $ `# of Wind Turbines`             <dbl> 12750, 371
```

readr::read_csv()

`col_types` describes individual variables by name using `cols()`

```
milk <- read_csv(here::here(
  'data', 'milk_production.csv'),
  col_types = cols(year = col_character()))

glimpse(milk)
```

```
#> Rows: 2,400
#> Columns: 4
#> $ region        <chr> "Northeast", "Northeast", "N
#> $ state         <chr> "Maine", "New Hampshire", "V
#> $ year          <chr> "1970", "1970", "1970", "197
#> $ milk_produced <dbl> 6.1900e+08, 3.5600e+08, 1.97
```

`readr::read_csv()`

`col_types` describes individual variables by name using `cols()`

Type	dplyr::glimpse()	readr::parse_*()	readr::col_*()
Logical	<lgl>	parse_logical()	col_logical()
Numeric	<int> *or* <dbl>	parse_number()	col_number()
Character	<chr>	parse_character()	col_character()
Factor	<fct>	parse_factor(levels)	col_factor(levels)
Date	<date>	parse_date(format)	col_date(format)

Other option: Edit types **after** reading in the data

```
wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.x
  mutate(
    Ranking = as.numeric(Ranking),
    `Equivalent Homes Powered` = as.numeric(
    `Total Investment ($ Millions)` = as.nu
    `# of Wind Turbines` = as.numeric(`# of

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ Ranking                        <dbl> 1
#> $ State                          <chr> "
#> $ `Installed Capacity (MW)`      <dbl> 2
#> $ `Equivalent Homes Powered`     <dbl> 6
#> $ `Total Investment ($ Millions)` <dbl> 4
#> $ `Wind Projects Online`         <dbl> 1
#> $ `# of Wind Turbines`           <dbl> 1
```

```
milk <- read_csv(here::here(
  'data', 'milk_production.csv')) %>%
  mutate(year = as.character(year))

glimpse(milk)
```

```
#> Rows: 2,400
#> Columns: 4
#> $ region        <chr> "Northeast", "North
#> $ state         <chr> "Maine", "New Hamps
#> $ year          <chr> "1970", "1970", "19
#> $ milk_produced <dbl> 6.1900e+08, 3.5600e
```

Week 9: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

BREAK

4. Re-coding variables

5. Dates

Renaming made easy

`janitor::clean_names()`


```
wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx'))

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ Ranking                          <chr> "1.0", "2.0",
#> $ State                            <chr> "TEXAS", "OKLA
#> $ `Installed Capacity (MW)`        <dbl> 23262, 7495, 7
#> $ `Equivalent Homes Powered`       <chr> "6235000.0", "
#> $ `Total Investment ($ Millions)`  <chr> "42000.0", "13
#> $ `Wind Projects Online`           <dbl> 136, 45, 107,
#> $ `# of Wind Turbines`             <chr> "12750.0", "37
```

Renaming made easy

`janitor::clean_names()`


```
library(janitor)

wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx')) %>%
  clean_names()

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ ranking                  <chr> "1.0", "2.0", "3.0",
#> $ state                    <chr> "TEXAS", "OKLAHOMA",
#> $ installed_capacity_mw    <dbl> 23262, 7495, 7312, 5
#> $ equivalent_homes_powered <chr> "6235000.0", "226800
#> $ total_investment_millions <chr> "42000.0", "13700.0"
#> $ wind_projects_online     <dbl> 136, 45, 107, 104, 3
#> $ number_of_wind_turbines  <chr> "12750.0", "3717.0",
```

Renaming made easy

`janitor::clean_names()`


```
library(janitor)

wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx')) %>%
  clean_names(case = 'lower_camel')

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ ranking                <chr> "1.0", "2.0", "3.0", "
#> $ state                  <chr> "TEXAS", "OKLAHOMA", "
#> $ installedCapacityMw    <dbl> 23262, 7495, 7312, 568
#> $ equivalentHomesPowered <chr> "6235000.0", "2268000.
#> $ totalInvestmentMillions <chr> "42000.0", "13700.0",
#> $ windProjectsOnline     <dbl> 136, 45, 107, 104, 35,
#> $ numberOfWindTurbines   <chr> "12750.0", "3717.0", "
```

Renaming made easy

`janitor::clean_names()`


```
library(janitor)

wind <- read_excel(here::here(
  'data', 'US_State_Wind_Energy_Facts_2018.xlsx')) %>%
  clean_names(case = 'screaming_snake')

glimpse(wind)
```

```
#> Rows: 50
#> Columns: 7
#> $ RANKING                  <chr> "1.0", "2.0", "3.0",
#> $ STATE                    <chr> "TEXAS", "OKLAHOMA",
#> $ INSTALLED_CAPACITY_MW    <dbl> 23262, 7495, 7312, 5
#> $ EQUIVALENT_HOMES_POWERED <chr> "6235000.0", "226800
#> $ TOTAL_INVESTMENT_MILLIONS <chr> "42000.0", "13700.0"
#> $ WIND_PROJECTS_ONLINE     <dbl> 136, 45, 107, 104, 3
#> $ NUMBER_OF_WIND_TURBINES  <chr> "12750.0", "3717.0",
```

`select()`: more powerful than you probably thought

Example: data on sleeping patterns of different mammals

```
glimpse(msleep)
```

```
#> Rows: 83
#> Columns: 11
#> $ name         <chr> "Cheetah", "Owl monkey", "Mounta
#> $ genus        <chr> "Acinonyx", "Aotus", "Aplodontia
#> $ vore         <chr> "carni", "omni", "herbi", "omni",
#> $ order        <chr> "Carnivora", "Primates", "Rodenti
#> $ conservation <chr> "lc", NA, "nt", "lc", "domestica
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4,
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333333, 0.6666667,
#> $ awake        <dbl> 11.90, 7.00, 9.60, 9.10, 20.00, 9
#> $ brainwt      <dbl> NA, 0.01550, NA, 0.00029, 0.42300
#> $ bodywt       <dbl> 50.000, 0.480, 1.350, 0.019, 600.
```

`select()`: more powerful than you probably thought

Use `select()` to choose which columns to **keep**

```
msleep %>%
  select(name:order, sleep_total:sleep_cycle) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 7
#> $ name        <chr> "Cheetah", "Owl monkey", "Mou
#> $ genus       <chr> "Acinonyx", "Aotus", "Aplodon
#> $ vore        <chr> "carni", "omni", "herbi", "om
#> $ order       <chr> "Carnivora", "Primates", "Roc
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9, 4.0,
#> $ sleep_rem   <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2,
#> $ sleep_cycle <dbl> NA, NA, NA, 0.1333333, 0.6666
```

Use `select()` to choose which columns to **drop**

```
msleep %>%
  select(-(name:order)) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 7
#> $ conservation <chr> "lc", NA, "nt", "l
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4,
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3,
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333
#> $ awake        <dbl> 11.90, 7.00, 9.60,
#> $ brainwt      <dbl> NA, 0.01550, NA, 0
#> $ bodywt       <dbl> 50.000, 0.480, 1.3
```

Select columns based on **partial column names**

Select columns that start with "sleep":

```
msleep %>%
  select(name, starts_with("sleep")) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 4
#> $ name        <chr> "Cheetah", "Owl monkey",
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9,
#> $ sleep_rem   <dbl> NA, 1.8, 2.4, 2.3, 0.7,
#> $ sleep_cycle <dbl> NA, NA, NA, 0.1333333, 0
```

Select columns that contain "eep" and end with "wt":

```
msleep %>%
  select(contains("eep"), ends_with("wt")) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 5
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9,
#> $ sleep_rem   <dbl> NA, 1.8, 2.4, 2.3, 0.7,
#> $ sleep_cycle <dbl> NA, NA, NA, 0.1333333, 0
#> $ brainwt     <dbl> NA, 0.01550, NA, 0.00029
#> $ bodywt      <dbl> 50.000, 0.480, 1.350, 0.
```

Select columns based on their **data type**

Select only numeric columns:

```
msleep %>%
    select_if(is.numeric) %>%
    glimpse()
```

```
#> Rows: 83
#> Columns: 6
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.
#> $ sleep_rem   <dbl> NA, 1.8, 2.4, 2.3, 0.
#> $ sleep_cycle <dbl> NA, NA, NA, 0.1333333
#> $ awake       <dbl> 11.90, 7.00, 9.60, 9.
#> $ brainwt     <dbl> NA, 0.01550, NA, 0.00
#> $ bodywt      <dbl> 50.000, 0.480, 1.350,
```

Select only character columns:

```
msleep %>%
    select_if(is.character) %>%
    glimpse()
```

```
#> Rows: 83
#> Columns: 5
#> $ name         <chr> "Cheetah", "Owl monk
#> $ genus        <chr> "Acinonyx", "Aotus",
#> $ vore         <chr> "carni", "omni", "he
#> $ order        <chr> "Carnivora", "Primat
#> $ conservation <chr> "lc", NA, "nt", "lc"
```

Use `select()` to **reorder** variables

```
msleep %>%
    select(everything()) %>%
    glimpse()
```

```
#> Rows: 83
#> Columns: 11
#> $ name         <chr> "Cheetah", "Owl mo
#> $ genus        <chr> "Acinonyx", "Aotus
#> $ vore         <chr> "carni", "omni", "
#> $ order        <chr> "Carnivora", "Prim
#> $ conservation <chr> "lc", NA, "nt", "l
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4,
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3,
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333
#> $ awake        <dbl> 11.90, 7.00, 9.60,
#> $ brainwt      <dbl> NA, 0.01550, NA, 0
#> $ bodywt       <dbl> 50.000, 0.480, 1.3
```

```
msleep %>%
    select(conservation, awake, everything()) %>%
    glimpse()
```

```
#> Rows: 83
#> Columns: 11
#> $ conservation <chr> "lc", NA, "nt", "lc", "domes
#> $ awake        <dbl> 11.90, 7.00, 9.60, 9.10, 20.
#> $ name         <chr> "Cheetah", "Owl monkey", "Mo
#> $ genus        <chr> "Acinonyx", "Aotus", "Aplodo
#> $ vore         <chr> "carni", "omni", "herbi", "o
#> $ order        <chr> "Carnivora", "Primates", "Ro
#> $ sleep_total  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0,
#> $ sleep_rem    <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2,
#> $ sleep_cycle  <dbl> NA, NA, NA, 0.1333333, 0.666
#> $ brainwt      <dbl> NA, 0.01550, NA, 0.00029, 0.
#> $ bodywt       <dbl> 50.000, 0.480, 1.350, 0.019,
```

Use `select()` to **rename** variables

Use `rename()` to just change the name

```
msleep %>%
  rename(
    animal = name,
    extinction_threat = conservation) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 11
#> $ animal            <chr> "Cheetah", "Owl mo
#> $ genus             <chr> "Acinonyx", "Aotus
#> $ vore              <chr> "carni", "omni", "
#> $ order             <chr> "Carnivora", "Prim
#> $ extinction_threat <chr> "lc", NA, "nt", "l
#> $ sleep_total       <dbl> 12.1, 17.0, 14.4,
#> $ sleep_rem         <dbl> NA, 1.8, 2.4, 2.3,
#> $ sleep_cycle       <dbl> NA, NA, NA, 0.1333
#> $ awake             <dbl> 11.90, 7.00, 9.60,
#> $ brainwt           <dbl> NA, 0.01550, NA, 0
#> $ bodywt            <dbl> 50.000, 0.480, 1.3
```

Use `select()` to change the name **and drop everything else**

```
msleep %>%
  select(
    animal = name,
    extinction_threat = conservation) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 2
#> $ animal            <chr> "Cheetah", "Owl mo
#> $ extinction_threat <chr> "lc", NA, "nt", "l
```

Use `select()` to **rename** variables

Use `rename()` to just change the name

```
msleep %>%
  rename(
    animal = name,
    extinction_threat = conservation) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 11
#> $ animal            <chr> "Cheetah", "Owl mo
#> $ genus             <chr> "Acinonyx", "Aotus
#> $ vore              <chr> "carni", "omni", "
#> $ order             <chr> "Carnivora", "Prim
#> $ extinction_threat <chr> "lc", NA, "nt", "l
#> $ sleep_total       <dbl> 12.1, 17.0, 14.4,
#> $ sleep_rem         <dbl> NA, 1.8, 2.4, 2.3,
#> $ sleep_cycle       <dbl> NA, NA, NA, 0.1333
#> $ awake             <dbl> 11.90, 7.00, 9.60,
#> $ brainwt           <dbl> NA, 0.01550, NA, 0
#> $ bodywt            <dbl> 50.000, 0.480, 1.3
```

Use `select()` + `everything()` to change names **and keep everything else**

```
msleep %>%
  select(
    animal = name,
    extinction_threat = conservation,
    everything()) %>%
  glimpse()
```

```
#> Rows: 83
#> Columns: 11
#> $ animal            <chr> "Cheetah", "Owl mo
#> $ extinction_threat <chr> "lc", NA, "nt", "l
#> $ genus             <chr> "Acinonyx", "Aotus
#> $ vore              <chr> "carni", "omni", "
#> $ order             <chr> "Carnivora", "Prim
#> $ sleep_total       <dbl> 12.1, 17.0, 14.4,
#> $ sleep_rem         <dbl> NA, 1.8, 2.4, 2.3,
#> $ sleep_cycle       <dbl> NA, NA, NA, 0.1333
#> $ awake             <dbl> 11.90, 7.00, 9.60,
```

Your turn

Read in the `hot_dog_winners.xlsx` file and adjust the variable names and types to the following:

```
#> Rows: 42
#> Columns: 7
#> $ year                <dbl> 1980,
#> $ competitor.mens     <chr> "Paul
#> $ competitor.womens   <chr> NA, NA
#> $ dogs_eaten.mens     <dbl> 9.10,
#> $ dogs_eaten.womens   <dbl> NA, NA
#> $ country.mens        <chr> "Unite
#> $ country.womens      <chr> NA, NA
```

15:00

	A	B	C	D	E	F	G
1	Year	Mens	Dogs eaten	Country	Womens	Dogs eaten	Country
2	1980	Paul Siederman & Joe Baldini	9.1	United States			
3	1981	Thomas DeBerry	11	United States			
4	1982	Steven Abrams	11	United States			
5	1983	Luis Llamas	19.5	Mexico			
6	1984	Birgit Felden	9.5	Germany			
7	1985	Oscar Rodriguez	11.75	United States			
8	1986	Mark Heller	15.5	United States			
9	1987	Don Wolfman	12	United States			
10	1988	Jay Green	14	United States			
11	1989	Jay Green	13	United States			
12	1990	Mike DeVito	16	United States			
13	1991	Frank Dellarosa	21.5*	United States			
14	1992	Frank Dellarosa	19	United States			
15	1993	Mike DeVito	17	United States			
16	1994	Mike DeVito	20	United States			
17	1995	Edward Krachie	19.5	United States			
18	1996	Edward Krachie	22.25*	United States			
19	1997	Hirofumi Nakajima	24.5*	Japan			
20	1998	Hirofumi Nakajima	19	Japan			
21	1999	Steve Keiner	20.25	United States			
22	2000	Kazutoyo Arai	25.13*	Japan			
23	2001	Takeru Kobayashi	50*	Japan			
24	2002	Takeru Kobayashi	50.5*	Japan			
25	2003	Takeru Kobayashi	44.5	Japan			
26	2004	Takeru Kobayashi	53.5*	Japan			
27	2005	Takeru Kobayashi	49	Japan			
28	2006	Takeru Kobayashi	53.75*	Japan			
29	2007	Joey Chestnut	66*	United States			
30	2008	Joey Chestnut	59	United States			
31	2009	Joey Chestnut	68*	United States			
32	2010	Joey Chestnut	54	United States			
33	2011	Joey Chestnut	62	United States	Sonya Thomas	40*	United States
34	2012	Joey Chestnut	68	United States	Sonya Thomas	45*	United States
35	2013	Joey Chestnut	69*	United States	Sonya Thomas	36.75	United States
36	2014	Joey Chestnut	61	United States	Miki Sudo	34	United States
37	2015	Matt Stonie	62	United States	Miki Sudo	38	United States
38	2016	Joey Chestnut	70*	United States	Miki Sudo	38.5	United States
39	2017	Joey Chestnut	72*	United States	Miki Sudo	41	United States
40	2018	Joey Chestnut	74*	United States	Miki Sudo	37	United States
41	2019	Joey Chestnut	71	United States	Miki Sudo	31	United States
42							
43	Notes: * means new record						

Break!

Stand up, Move around, Stretch!

05:00

Week 9: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

BREAK

4. Re-coding variables

5. Dates

Recoding with `ifelse()`

Example: Create a variable, `cost_high`, that is `TRUE` if the repair costs were greater than the median costs and `FALSE` otherwise.

```
wildlife_impacts1 <- wildlife_impacts %>%
  rename(cost = cost_repairs_infl_adj) %>%
  filter(!is.na(cost)) %>%
  mutate(
    cost_median = median(cost),
    cost_high = ifelse(cost > cost_median, TRUE, FALSE))

wildlife_impacts1 %>%
  select(cost, cost_median, cost_high) %>%
  head()
```

```
#> # A tibble: 6 x 3
#>     cost cost_median cost_high
#>    <dbl>       <dbl> <lgl>
#> 1   1000       26783 FALSE
#> 2    200       26783 FALSE
#> 3  10000       26783 FALSE
#> 4 100000       26783 TRUE
#> 5  20000       26783 FALSE
#> 6 487000       26783 TRUE
```

Recoding with **nested** `ifelse()`

Create a variable, `season`, based on the `incident_month` variable.

```
wildlife_impacts2 <- wildlife_impacts %>%
  mutate(season = ifelse(
    incident_month %in% c(3, 4, 5), 'spring', ifelse(
    incident_month %in% c(6, 7, 8), 'summer', ifelse(
    incident_month %in% c(9, 10, 11), 'fall', 'winter')))
  )

wildlife_impacts2 %>%
  distinct(incident_month, season) %>%
  head()
```

```
#> # A tibble: 6 x 2
#>   incident_month season
#>            <dbl> <chr>
#> 1             12 winter
#> 2             11 fall
#> 3             10 fall
#> 4              9 fall
#> 5              8 summer
#> 6              7 summer
```

Recoding with `case_when()`

Create a variable, `season`, based on the `incident_month` variable.

Note: If you don't include the final `TRUE ~ 'winter'` condition, you'll get `NA` for those cases.

```
wildlife_impacts2 <- wildlife_impacts %>%
  mutate(season = case_when(
    incident_month %in% c(3, 4, 5) ~ 'spring',
    incident_month %in% c(6, 7, 8) ~ 'summer',
    incident_month %in% c(9, 10, 11) ~ 'fall',
    TRUE ~ 'winter')
  )

wildlife_impacts2 %>%
  distinct(incident_month, season) %>%
  head()
```

```
#> # A tibble: 6 x 2
#>   incident_month season
#>            <dbl> <chr>
#> 1             12 winter
#> 2             11 fall
#> 3             10 fall
#> 4              9 fall
#> 5              8 summer
#> 6              7 summer
```

Recoding with `case_when()` with `between()`

Create a variable, `season`, based on the `incident_month` variable.

```
wildlife_impacts2 <- wildlife_impacts %>%
  mutate(season = case_when(
    between(incident_month, 3, 5) ~ 'spring',
    between(incident_month, 6, 8) ~ 'summer',
    between(incident_month, 9, 11) ~ 'fall',
    TRUE ~ 'winter')
  )

wildlife_impacts2 %>%
    distinct(incident_month, season) %>%
    head()
```

```
#> # A tibble: 6 x 2
#>   incident_month season
#>            <dbl> <chr>
#> 1             12 winter
#> 2             11 fall
#> 3             10 fall
#> 4              9 fall
#> 5              8 summer
#> 6              7 summer
```

`case_when()` is "cleaner" than `ifelse()`

Convert the `num_engs` variable into a word of the number.

`ifelse()`

```
wildlife_impacts3 <- wildlife_impacts %>%
  mutate(num_engs = ifelse(
    num_engs == 1, 'one', ifelse(
    num_engs == 2, 'two', ifelse(
    num_engs == 3, 'three', ifelse(
    num_engs == 4, 'four',
    as.character(num_engs)))))
  )

unique(wildlife_impacts3$num_engs)
```

```
#> [1] "two"   NA      "three" "four"  "one"
```

`case_when()`

```
wildlife_impacts3 <- wildlife_impacts %>%
  mutate(num_engs = case_when(
    num_engs == 1 ~ 'one',
    num_engs == 2 ~ 'two',
    num_engs == 3 ~ 'three',
    num_engs == 4 ~ 'four')
  )

unique(wildlife_impacts3$num_engs)
```

```
#> [1] "two"   NA      "three" "four"  "on
```

Break a single variable into two with `separate()`

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/2059536
#> 3 Brazil       1999 37737/172006
#> 4 Brazil       2000 80488/174504
#> 5 China        1999 212258/12729
#> 6 China        2000 213766/12804
```

```
tb_rates %>%
  separate(rate, into = c("cases", "population"))
```

```
#> # A tibble: 6 x 4
#>   country      year cases  population
#>   <chr>       <int> <chr>  <chr>
#> 1 Afghanistan  1999 745    19987071
#> 2 Afghanistan  2000 2666   20595360
#> 3 Brazil       1999 37737  172006362
#> 4 Brazil       2000 80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Break a single variable into two with `separate()`

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/2059536
#> 3 Brazil       1999 37737/172006
#> 4 Brazil       2000 80488/174504
#> 5 China        1999 212258/12729
#> 6 China        2000 213766/12804
```

```
tb_rates %>%
  separate(rate, into = c("cases", "population"),
           sep = "/")
```

```
#> # A tibble: 6 x 4
#>   country      year cases  population
#>   <chr>       <int> <chr>  <chr>
#> 1 Afghanistan  1999 745    19987071
#> 2 Afghanistan  2000 2666   20595360
#> 3 Brazil       1999 37737  172006362
#> 4 Brazil       2000 80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

Break a single variable into two with `separate()`

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/2059536
#> 3 Brazil       1999 37737/172006
#> 4 Brazil       2000 80488/174504
#> 5 China        1999 212258/12729
#> 6 China        2000 213766/12804
```

```
tb_rates %>%
  separate(rate, into = c("cases", "population"),
           sep = "/",
           convert = TRUE)
```

```
#> # A tibble: 6 x 4
#>   country      year  cases population
#>   <chr>       <int>  <int>      <int>
#> 1 Afghanistan  1999    745   19987071
#> 2 Afghanistan  2000   2666   20595360
#> 3 Brazil       1999  37737  172006362
#> 4 Brazil       2000  80488  174504898
#> 5 China        1999 212258 1272915272
#> 6 China        2000 213766 1280428583
```

You can also break up a variable by an index

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/20595360
#> 3 Brazil       1999 37737/172006362
#> 4 Brazil       2000 80488/174504898
#> 5 China        1999 212258/1272915272
#> 6 China        2000 213766/1280428583
```

```
tb_rates %>%
  separate(year, into = c("century", "year"),
           sep = 2)
```

```
#> # A tibble: 6 x 4
#>   country     century year  rate
#>   <chr>       <chr>   <chr> <chr>
#> 1 Afghanistan 19      99    745/19987071
#> 2 Afghanistan 20      00    2666/20595360
#> 3 Brazil      19      99    37737/172006362
#> 4 Brazil      20      00    80488/174504898
#> 5 China       19      99    212258/1272915272
#> 6 China       20      00    213766/1280428583
```

`unite()`: The opposite of `separate()`

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/2059536
#> 3 Brazil       1999 37737/172006
#> 4 Brazil       2000 80488/174504
#> 5 China        1999 212258/12729
#> 6 China        2000 213766/12804
```

```
tb_rates %>%
  separate(year, into = c("century", "year"),
           sep = 2) %>%
  unite(year_new, century, year)
```

```
#> # A tibble: 6 x 3
#>   country     year_new rate
#>   <chr>       <chr>    <chr>
#> 1 Afghanistan 19_99    745/19987071
#> 2 Afghanistan 20_00    2666/20595360
#> 3 Brazil      19_99    37737/172006362
#> 4 Brazil      20_00    80488/174504898
#> 5 China       19_99    212258/1272915272
#> 6 China       20_00    213766/1280428583
```

`unite()`: The opposite of `separate()`

```
tb_rates
```

```
#> # A tibble: 6 x 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/19987071
#> 2 Afghanistan  2000 2666/2059536
#> 3 Brazil       1999 37737/172006
#> 4 Brazil       2000 80488/174504
#> 5 China        1999 212258/12729
#> 6 China        2000 213766/12804
```

```
tb_rates %>%
  separate(year, into = c("century", "year"),
           sep = 2) %>%
  unite(year_new, century, year,
        sep = "")
```

```
#> # A tibble: 6 x 3
#>   country     year_new rate
#>   <chr>       <chr>    <chr>
#> 1 Afghanistan 1999     745/19987071
#> 2 Afghanistan 2000     2666/20595360
#> 3 Brazil      1999     37737/172006362
#> 4 Brazil      2000     80488/174504898
#> 5 China       1999     212258/1272915272
#> 6 China       2000     213766/1280428583
```

Week 9: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

BREAK

4. Re-coding variables

5. Dates


What is your idea of a perfect date?
YYYY/MM/DD, other formats can be really confusing
You will die single.
?

Create dates from strings - **order is the ONLY thing that matters!**

Year-Month-Day

```
ymd('2020-02-26')
```

```
#> [1] "2020-02-26"
```

Create dates from strings - **order is the ONLY thing that matters!**

Year-Month-Day

```
ymd('2020-02-26')
```

```
#> [1] "2020-02-26"
```

```
ymd('2020 Feb 26')
```

```
#> [1] "2020-02-26"
```

Create dates from strings - **order is the ONLY thing that matters!**

Year-Month-Day

```
ymd('2020-02-26')
#> [1] "2020-02-26"
ymd('2020 Feb 26')
#> [1] "2020-02-26"
ymd('2020 Feb. 26')
#> [1] "2020-02-26"
ymd('2020 february 26')
#> [1] "2020-02-26"
```

Month-Day-Year

```
mdy('February 26, 2020')
#> [1] "2020-02-26"
mdy('Feb. 26, 2020')
#> [1] "2020-02-26"
mdy('Feb 26 2020')
#> [1] "2020-02-26"
```

Day-Month-Year

```
dmy('26 February 2020')
#> [1] "2020-02-26"
dmy('26 Feb. 2020')
#> [1] "2020-02-26"
dmy('26 Feb, 2020')
#> [1] "2020-02-26"
```

Check out the `lubridate` **cheat sheet**

Extracting information from dates

```
date <- today()
date
```

```
#> [1] "2021-03-09"
```

```
# Get the year
year(date)
```

```
#> [1] 2021
```

Extracting information from dates

```
date <- today()
date
```

```
#> [1] "2021-03-09"
```

```
# Get the year
year(date)
```

```
#> [1] 2021
```

```
# Get the month
month(date)
```

```
#> [1] 3
```

```
# Get the month name
month(date, label = TRUE, abbr = FALSE)
```

```
#> [1] March
#> Levels: January < February < March < April < May < J
```

```
# Get the day
day(date)
```

```
#> [1] 9
```

```
# Get the weekday
wday(date)
```

```
#> [1] 3
```

```
# Get the weekday name
wday(date, label = TRUE, abbr = TRUE)
```

```
#> [1] Tue
#> Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

Quick practice

On what day of the week were you born?

```
wday("2020-02-26", label = TRUE)
```

```
#> [1] Wed
#> Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

Modifying date elements

```
date <- today()
date
```

```
#> [1] "2021-03-09"
```

```
# Change the year
year(date) <- 2016
date
```

```
#> [1] "2016-03-09"
```

```
# Change the day
day(date) <- 30
```

```
date
```

```
#> [1] "2016-03-30"
```

Quick practice

What do you think will happen if we do this?

```
date <- ymd("2021-02-28")
day(date) <- 30
```

```
date
```

```
#> [1] "2021-03-02"
```

Your turn

20:00

1) Use `case_when()` to modify the `phase_of_flt` variable in the `wildlife_impacts` data:

- The values `'approach'`, `'arrival'`, `'descent'`, and `'landing roll'` should be merged into a single value called `'arrival'`.
- The values `'climb'`, `'departure'`, and `'take-off run'` should be merged into a single value called `'departure'`.
- All other values should be called `'other'`.

Before:

```
unique(str_to_lower(wildlife_impacts$phase_of_flt))
```

```
#>  [1] "climb"        "landing roll" NA             "apprc
```

After:

```
#> [1] "departure" "arrival"   "other"
```

2) Use the **lubridate** package to create a new variable, `weekday_name`, from the `incident_date` variable in the `wildlife_impacts` data.

3) Use `weekday_name` and `phase_of_flt` to make this plot of "arrival" and "departure" impacts from **Mar. 2016**.