# Week 3: *Cleaning Data*

🏛 EMSE 4575: Exploratory Data Analysis

👤 John Paul Helveston

📅 September 14, 2022

# *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:

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

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

# Week 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# Week 3: *Cleaning Data*

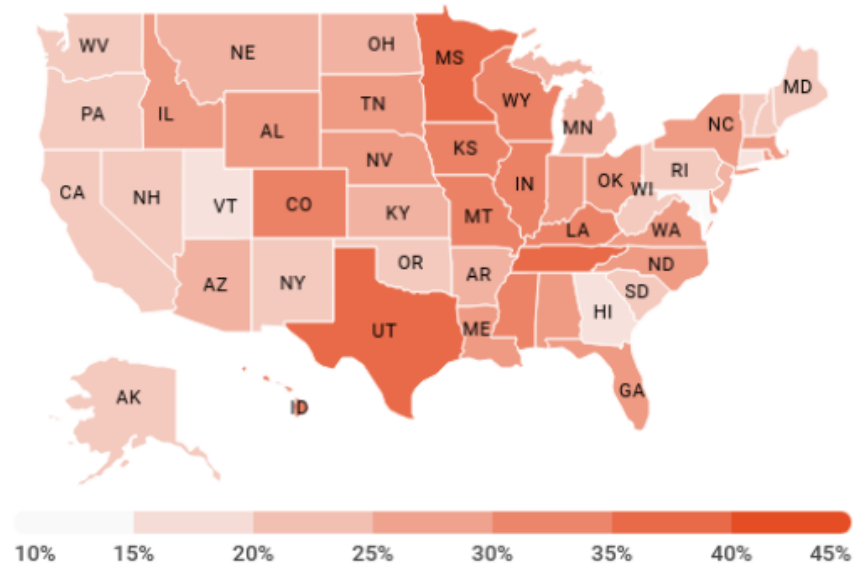1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

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()

band_members

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

band_instruments

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



inner_join(x, y)
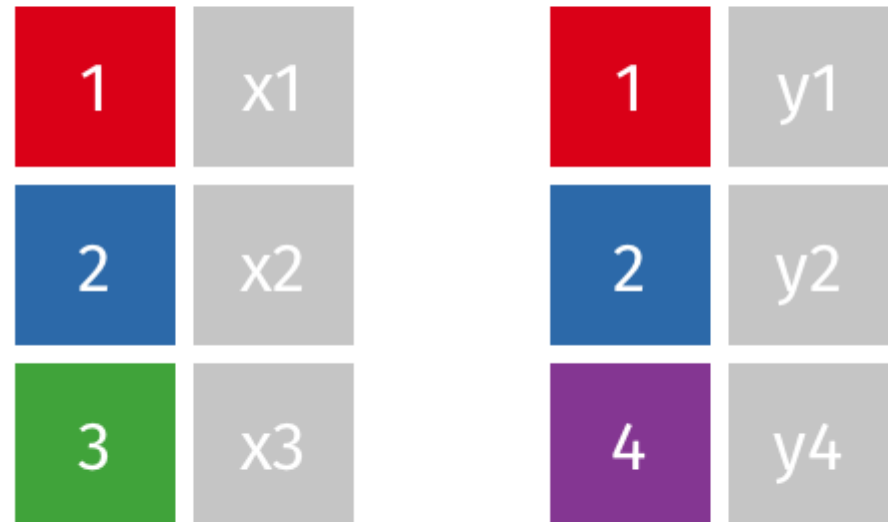
# full_join()

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

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



full_join(x, y)

# left_join()

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

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



right_join(x, y)

# Specify the joining variable name

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

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

```
#> # A tibble: 3 × 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```
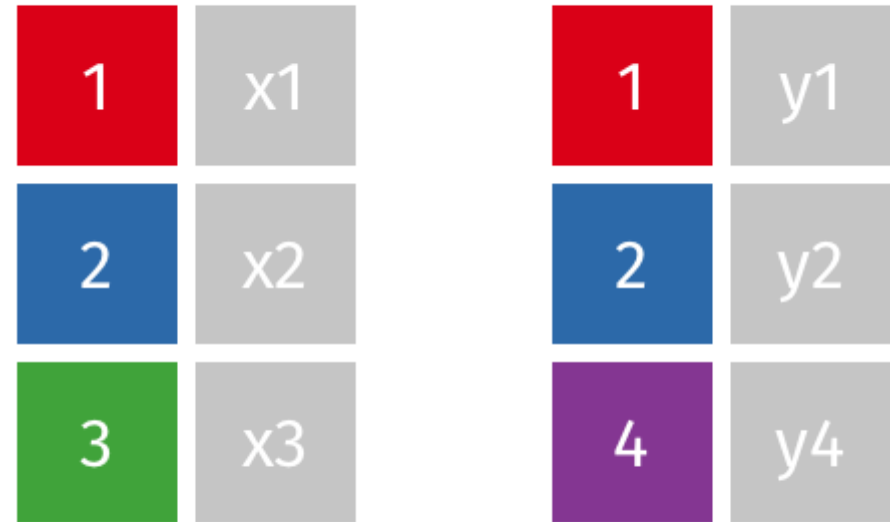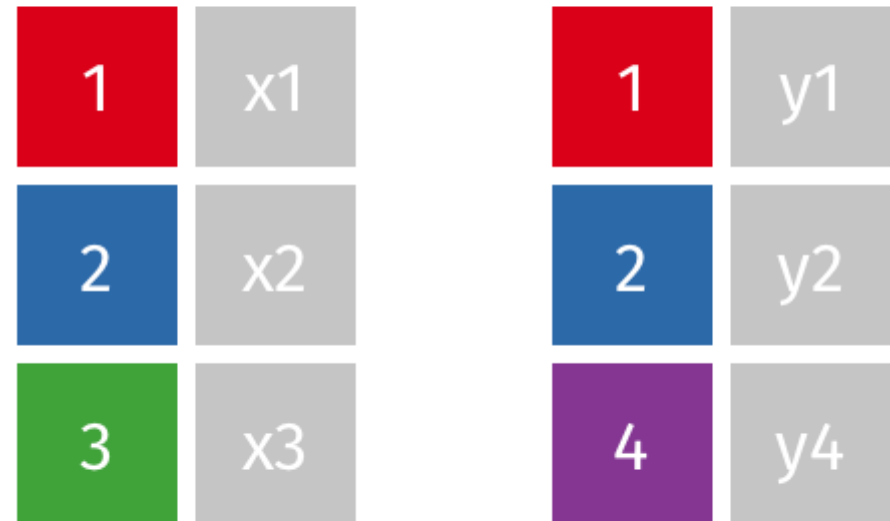
```
band_members %>%
    left_join(band_instruments,
              by = 'name')
```

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

```
band_instruments2
```

```
#> # A tibble: 3 × 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 × 3
#>   name  band    plays
#>   <chr> <chr>   <chr>
#> 1 Mick  Stones  <NA>
#> 2 John  Beatles guitar
#> 3 Paul  Beatles bass
```
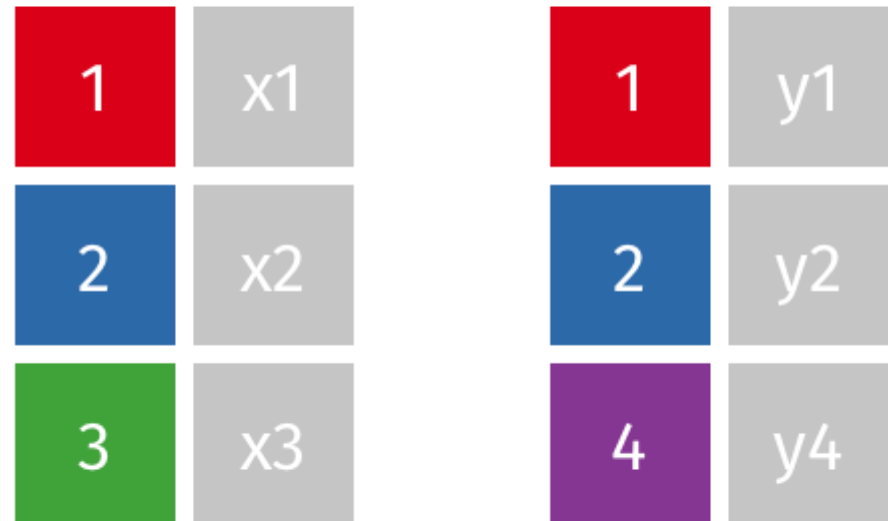
# Specify the joining variable name

Or just rename the joining variable in a pipe

```
band_members
```

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

```
band_instruments2
```

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

# Your turn

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 × 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,
#> $ airport_id          <chr> "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "
#> $ airport             <chr> "PORTLAND INTL JETPORT (ME)", "PORTLAND INTL JETP
#> $ operator            <chr> "AMERICAN AIRLINES", "AMERICAN AIRLINES", "AMERIC
#> $ 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",
#> $ 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, 2
#> $ 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", "ar
#> $ 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 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# 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

```r
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", "\
#> $ 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

```r
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.num
    `# 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
```

```r
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 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# 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()



```r
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()`



```r
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()

```r
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", "Mountai
#> $ genus        <chr> "Acinonyx", "Aotus", "Aplodontia"
#> $ vore         <chr> "carni", "omni", "herbi", "omni",
#> $ order        <chr> "Carnivora", "Primates", "Rodenti
#> $ conservation <chr> "lc", NA, "nt", "lc", "domesticat
#> $ 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", "Aplodor
#> $ vore        <chr> "carni", "omni", "herbi", "on
#> $ 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
```

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | Year | Mens | Dogs eaten | Country | Womens | Dogs | |
| 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 | | | | | | |

# Quiz 1

Link is in the #class channel



```
10:00
```
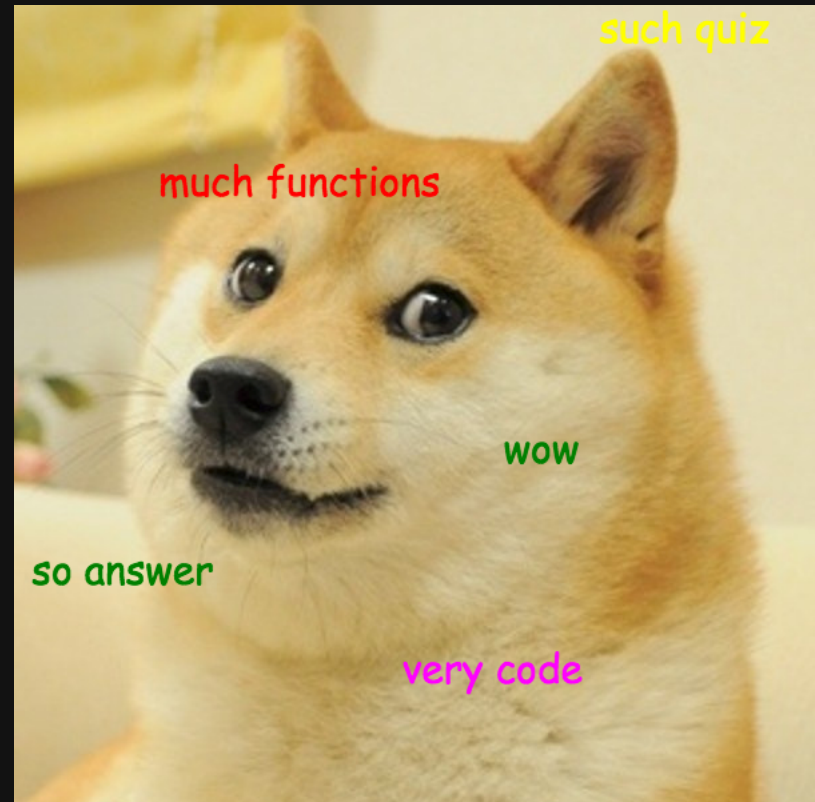
# Week 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# 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.

```r
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 × 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.

```r
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 × 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.

```r
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 × 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 × 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()

```r
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()

```r
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 × 3
#>   country        year rate
#> * <chr>         <int> <chr>
#> 1 Afghanistan    1999 745/1998707
#> 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 × 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 × 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 × 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 × 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/199870711
#> 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 × 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 × 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/1998707
#> 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)
```

```
#> # A tibble: 6 × 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 × 3
#>   country      year rate
#> * <chr>       <int> <chr>
#> 1 Afghanistan  1999 745/199870711
#> 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 × 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 × 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 × 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
```
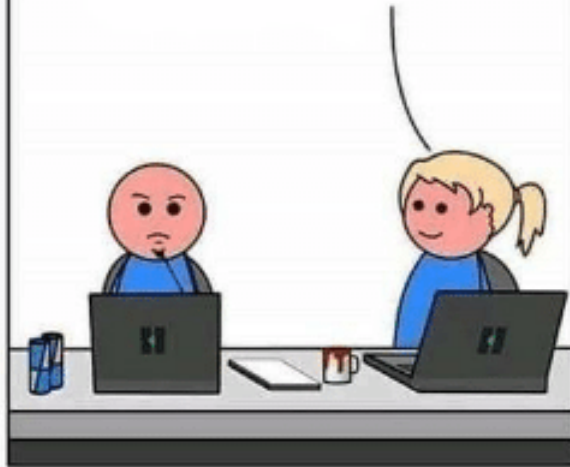
# Week 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

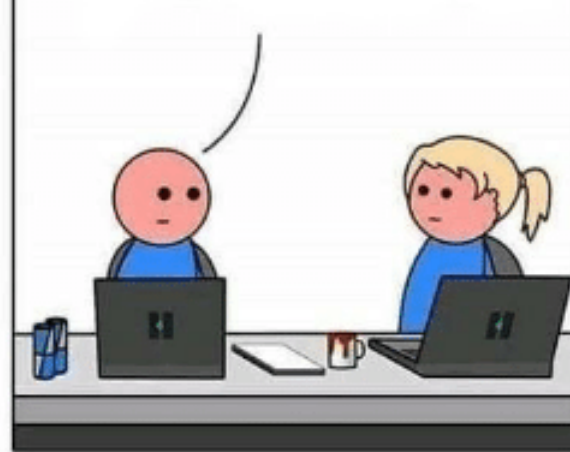3. Are your variables the right *name*?

QUIZ 1

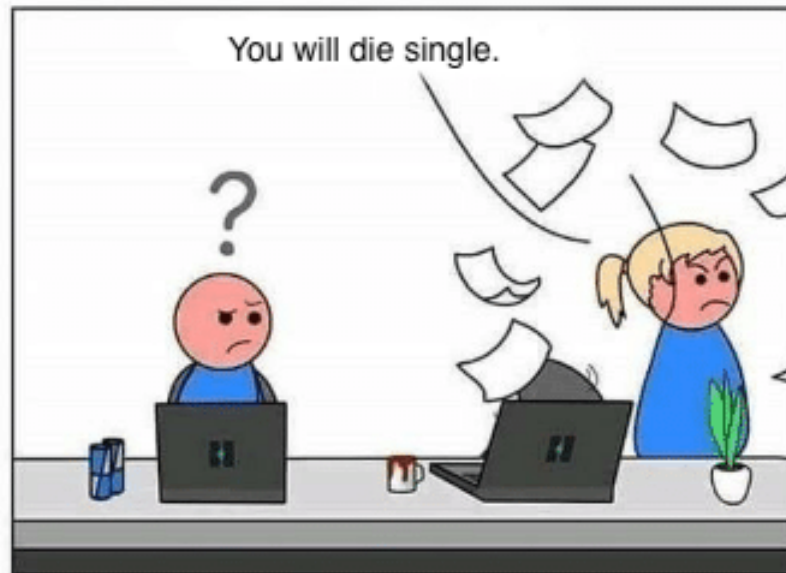4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# 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 | Month-Day-Year | Day-Month-Year |
|---|---|---|
| `ymd('2020-02-26')` | `mdy('February 26, 2020')` | `dmy('26 February 2020')` |
| `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` |
| `ymd('2020 Feb 26')` | `mdy('Feb. 26, 2020')` | `dmy('26 Feb. 2020')` |
| `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` |
| `ymd('2020 Feb. 26')` | `mdy('Feb 26 2020')` | `dmy('26 Feb, 2020')` |
| `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` | `#> [1] "2020-02-26"` |
| `ymd('2020 february 26')` | | |
| `#> [1] "2020-02-26"` | | |

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

# Extracting information from dates

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

```
#> [1] "2022-09-13"
```

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

```
#> [1] 2022
```

# Extracting information from dates

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

```
#> [1] "2022-09-13"
```

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

```
#> [1] 2022
```

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

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

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

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

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

```
#> [1] 13
```

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

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

```r
# 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] "2022-09-13"
```

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

```
#> [1] "2016-09-13"
```

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

```
date
```

```
#> [1] "2016-09-30"
```

# Quick practice

What do you think will happen if we do this?

```
date <- ymd("2022-02-28")
day(date) <- 30
```

```
date
```

```
#> [1] "2022-03-02"
```

# Your turn

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:

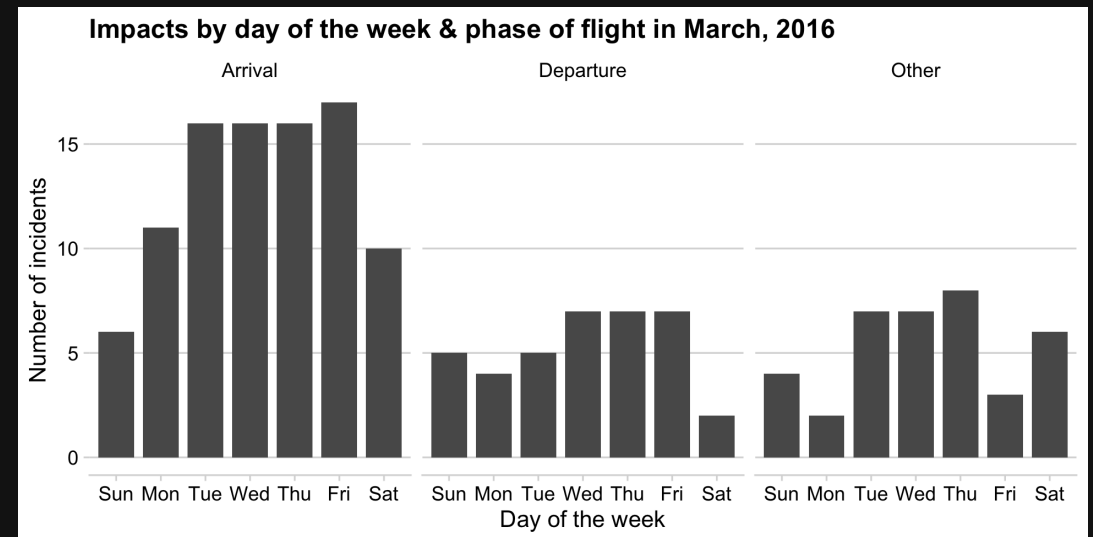```
unique(str_to_lower(wildlife_impacts$phase_of_flt))
```

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

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**.



Impacts by day of the week & phase of flight in March, 2016

# Week 3: *Cleaning Data*

1. Merging datasets with joins

2. Are your variables the right *type*?

3. Are your variables the right *name*?

QUIZ 1

4. Re-coding variables

5. Dates

6. Dealing with messy Excel files

# When columns are repeated

Example: Winners of Nathan's hot dog eating contest

# Stragies

## 1. Divide & conquer

## 2. Gather, separate, spread

| | 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 | | | | | | |

# Strategy 1: divide & conquer

Steps:

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table

```r
hot_dogs <- read_excel(
    here::here('data', 'hot_dog_winners.xlsx'),
    sheet = 'hot_dog_winners') %>%
    clean_names() %>%
    dplyr::filter(!is.na(mens))

glimpse(hot_dogs)
```

```
#> Rows: 40
#> Columns: 7
#> $ year         <chr> "1980", "1981", "1982", "1983
#> $ mens         <chr> "Paul Siederman & Joe Baldini
#> $ dogs_eaten_3 <chr> "9.1", "11", "11", "19.5", "9
#> $ country_4    <chr> "United States", "United Stat
#> $ womens       <chr> NA, NA, NA, NA, NA, NA, NA, N
#> $ dogs_eaten_6 <chr> NA, NA, NA, NA, NA, NA, NA, N
#> $ country_7    <chr> NA, NA, NA, NA, NA, NA, NA, N
```

# Strategy 1: divide & conquer

Steps

1. Read in the data
2. Clean the names
3. Remove ∗ note at bottom of table
4. **Split data into two competitions with the same variable names**
5. **Create new variable in each data frame: `competition`**

```
hot_dogs_m <- hot_dogs %>%
    select(
        year,
        competitor = mens,
        dogs_eaten = dogs_eaten_3,
        country    = country_4) %>%
    mutate(competition = 'Mens')

hot_dogs_w <- hot_dogs %>%
    select(
        year,
        competitor = womens,
        dogs_eaten = dogs_eaten_6,
        country    = country_7) %>%
    mutate(competition = 'Womens') %>%
    dplyr::filter(!is.na(competitor))
```

# Strategy 1: divide & conquer

Steps

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table
4. Split data into two competitions with the same variable names
5. Create new variable in each data frame: `competition`
6. **Merge data together with `bind_rows()`**
7. **Clean up final data frame**

```r
hot_dogs <- bind_rows(hot_dogs_m, hot_dogs_w) %>%
    mutate(
        new_record = str_detect(dogs_eaten, "\\*"),
        dogs_eaten = parse_number(dogs_eaten),
        year       = as.numeric(year))

glimpse(hot_dogs)
```

```
#> Rows: 49
#> Columns: 6
#> $ year        <dbl> 1980, 1981, 1982, 1983, 1984,
#> $ competitor  <chr> "Paul Siederman & Joe Baldini"
#> $ dogs_eaten  <dbl> 9.10, 11.00, 11.00, 19.50, 9.5
#> $ country     <chr> "United States", "United State
#> $ competition <chr> "Mens", "Mens", "Mens", "Mens"
#> $ new_record  <lgl> FALSE, FALSE, FALSE, FALSE, FA
```

| | 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 | | | | | | |

```
head(hot_dogs)
```

```
#> # A tibble: 6 × 6
#>    year competitor                     dogs_eaten country      competit
#>   <dbl> <chr>                               <dbl> <chr>        <chr>
#> 1  1980 Paul Siederman & Joe Baldini          9.1 United States Mens
#> 2  1981 Thomas DeBerry                       11   United States Mens
#> 3  1982 Steven Abrams                        11   United States Mens
#> 4  1983 Luis Llamas                          19.5 Mexico        Mens
#> 5  1984 Birgit Felden                         9.5 Germany       Mens
#> 6  1985 Oscar Rodriguez                      11.8 United States Mens
```

# Strategy 2: gather, separate, spread

Steps:

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table

```
hot_dogs <- read_excel(
    here::here('data', 'hot_dog_winners.xlsx'),
    sheet = 'hot_dog_winners') %>%
    clean_names() %>%
    dplyr::filter(!is.na(mens))

glimpse(hot_dogs)
```

```
#> Rows: 40
#> Columns: 7
#> $ year        <chr> "1980", "1981", "1982", "198
#> $ mens        <chr> "Paul Siederman & Joe Baldini
#> $ dogs_eaten_3 <chr> "9.1", "11", "11", "19.5", "9
#> $ country_4    <chr> "United States", "United Stat
#> $ womens      <chr> NA, NA, NA, NA, NA, NA, NA, N
#> $ dogs_eaten_6 <chr> NA, NA, NA, NA, NA, NA, NA, N
#> $ country_7    <chr> NA, NA, NA, NA, NA, NA, NA, N
```

# Strategy 2: gather, separate, spread

Steps:

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table
4. **Rename variables**
5. **Gather all the "joint" variables**

```
hot_dogs <- hot_dogs %>%
    select(
        year,
        competitor.mens   = mens,
        competitor.womens = womens,
        dogs_eaten.mens   = dogs_eaten_3,
        dogs_eaten.womens = dogs_eaten_6,
        country.mens      = country_4,
        country.womens    = country_7) %>%
    gather(key = 'variable', value = 'value',
           competitor.mens:country.womens)

head(hot_dogs, 3)
```

```
#> # A tibble: 3 × 3
#>    year  variable        value
#>    <chr> <chr>           <chr>
#> 1 1980  competitor.mens Paul Siederman & Joe Baldi
#> 2 1981  competitor.mens Thomas DeBerry
#> 3 1982  competitor.mens Steven Abrams
```

# Strategy 2: gather, separate, spread

Steps:

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table
4. Rename variables
5. Gather all the "joint" variables
6. **Separate "joint" variables into components**

```r
hot_dogs <- hot_dogs %>%
    separate(variable, into = c('variable', 'competition'),
             sep = '\\.')

head(hot_dogs)
```

```
#> # A tibble: 6 × 4
#>    year  variable   competition value
#>    <chr> <chr>      <chr>       <chr>
#> 1 1980  competitor mens        Paul Siederman & Joe Baldini
#> 2 1981  competitor mens        Thomas DeBerry
#> 3 1982  competitor mens        Steven Abrams
#> 4 1983  competitor mens        Luis Llamas
#> 5 1984  competitor mens        Birgit Felden
#> 6 1985  competitor mens        Oscar Rodriguez
```

# Strategy 2: gather, separate, spread

Steps:

1. Read in the data
2. Clean the names
3. Remove * note at bottom of table
4. Rename variables
5. Gather all the "joint" variables
6. Separate "joint" variables into components
7. **Spread variable and value back to columns**

```
hot_dogs <- hot_dogs %>%
    spread(key = variable, value = value) %>%
    mutate(
        new_record = str_detect(dogs_eaten, "\\*"),
        dogs_eaten = parse_number(dogs_eaten),
        year       = as.numeric(year))

glimpse(hot_dogs)
```

```
#> Rows: 80
#> Columns: 6
#> $ year        <dbl> 1980, 1980, 1981, 1981, 1982, 1982, 198
#> $ competition <chr> "mens", "womens", "mens", "womens", "me
#> $ competitor  <chr> "Paul Siederman & Joe Baldini", NA, "Th
#> $ country     <chr> "United States", NA, "United States", N
#> $ dogs_eaten  <dbl> 9.10, NA, 11.00, NA, 11.00, NA, 19.50,
#> $ new_record  <lgl> FALSE, NA, FALSE, NA, FALSE, NA, FALSE,
```

## Divide & conquer

```r
hot_dogs <- read_excel(
    here::here('data', 'hot_dog_winners.xlsx'),
    sheet = 'hot_dog_winners') %>%
    clean_names() %>%
    dplyr::filter(!is.na(mens))

# Divide
hot_dogs_m <- hot_dogs %>%
    select(
        year,
        competitor = mens,
        dogs_eaten = dogs_eaten_3,
        country    = country_4) %>%
    mutate(competition = 'Mens')
hot_dogs_w <- hot_dogs %>%
    select(
        year,
        competitor = womens,
        dogs_eaten = dogs_eaten_6,
        country    = country_7) %>%
    mutate(competition = 'Womens') %>%
    dplyr::filter(!is.na(competitor))

# Merge and finish cleaning
hot_dogs <- bind_rows(hot_dogs_m, hot_dogs_w) %>%
    mutate(
        new_record = str_detect(dogs_eaten, "\\*"),
        dogs_eaten = parse_number(dogs_eaten),
        year       = as.numeric(year))
```

## Gather, separate, spread

```r
hot_dogs <- read_excel(
    here::here('data', 'hot_dog_winners.xlsx'),
    sheet = 'hot_dog_winners') %>%
    clean_names() %>%
    dplyr::filter(!is.na(mens)) %>%

# Rename variables
select(
    year,
    competitor.mens   = mens,
    competitor.womens = womens,
    dogs_eaten.mens   = dogs_eaten_3,
    dogs_eaten.womens = dogs_eaten_6,
    country.mens      = country_4,
    country.womens    = country_7) %>%
# Gather "joint" variables
gather(key = 'variable', value = 'value',
       competitor.mens:country.womens) %>%
# Separate "joint" variables
separate(variable, into = c('variable', 'competiti
        sep = '\\.') %>%
# Spread "joint" variables
spread(key = variable, value = value) %>%
# Finish cleaning
mutate(
    new_record = str_detect(dogs_eaten, "\\*"),
    dogs_eaten = parse_number(dogs_eaten),
    year       = as.numeric(year))
```

# Strategies for dealing with **sub-headers**

Example:

OICA passenger car sales data

**NEW PC REGISTRATIONS OR SALES**

*Estimated figures*

| REGIONS/COUNTRIES | 2005 | 2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013 | 2014 | 2015 | 2016 | 2017 | 2018 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| EUROPE | 17,906,455 | 18,685,556 | 19,618,588 | 18,821,599 | 16,608,761 | 16,499,863 | 17,167,600 | 16,191,269 | 15,942,273 | 16,154,279 | 16,410,563 | 17,291,819 | 17,974,281 | 17,912,336 |
| EU 28 countries + EFTA | 15,622,035 | 15,961,138 | 16,147,274 | 14,911,880 | 14,533,115 | 13,830,694 | 13,642,659 | 12,567,903 | 12,344,415 | 13,061,461 | 14,287,881 | 15,160,239 | 15,631,283 | 15,626,509 |
| EU 15 countries + EFTA | 14,565,695 | 14,820,182 | 14,842,186 | 13,602,038 | 13,668,808 | 12,984,549 | 12,815,435 | 11,773,281 | 11,555,153 | 12,148,648 | 13,261,258 | 13,971,468 | 14,320,223 | 14,210,016 |
| AUSTRIA | 307,915 | 308,594 | 298,182 | 293,697 | 319,403 | 328,563 | 356,145 | 336,010 | 319,035 | 303,318 | 308,555 | 329,604 | 353,320 | 341,068 |
| BELGIUM | 480,088 | 526,141 | 524,795 | 535,947 | 476,194 | 547,340 | 572,211 | 486,737 | 486,065 | 482,939 | 501,066 | 539,519 | 546,558 | 549,632 |
| DENMARK | 148,819 | 156,936 | 162,686 | 150,199 | 112,454 | 153,858 | 170,036 | 170,763 | 182,086 | 189,055 | 207,717 | 222,924 | 221,821 | 218,566 |
| FINLAND | 148,161 | 145,700 | 125,608 | 139,669 | 90,574 | 111,968 | 126,123 | 111,251 | 103,455 | 106,237 | 108,819 | 118,991 | 120,480 | 120,480 |
| FRANCE | 2,118,042 | 2,045,745 | 2,109,672 | 2,091,369 | 2,302,398 | 2,251,669 | 2,204,229 | 1,898,760 | 1,790,456 | 1,795,885 | 1,917,226 | 2,015,177 | 2,110,748 | 2,173,481 |
| GERMANY | 3,319,259 | 3,467,961 | 3,148,163 | 3,090,040 | 3,807,175 | 2,916,259 | 3,173,634 | 3,082,504 | 2,952,431 | 3,036,773 | 3,206,042 | 3,351,607 | 3,441,262 | 3,435,778 |
| GREECE | 269,728 | 267,669 | 279,745 | 267,295 | 219,730 | 141,501 | 97,680 | 58,482 | 58,694 | 71,218 | 75,805 | 78,873 | 88,083 | 103,431 |
| ICELAND | 18,060 | 17,129 | 15,942 | 9,033 | 2,113 | 3,106 | 5,038 | 7,902 | 7,274 | 9,537 | 14,004 | 18,442 | 21,324 | 17,976 |
| IRELAND | 171,742 | 178,484 | 186,325 | 151,607 | 57,453 | 88,446 | 89,911 | 79,498 | 74,367 | 96,284 | 124,804 | 146,600 | 131,332 | 125,557 |
| ITALY | 2,244,108 | 2,335,462 | 2,494,115 | 2,161,359 | 2,159,465 | 1,961,580 | 1,749,740 | 1,403,010 | 1,304,648 | 1,360,578 | 1,575,737 | 1,824,968 | 1,970,497 | 1,910,025 |
| LUXEMBOURG | 48,517 | 50,837 | 51,332 | 52,359 | 47,265 | 49,726 | 49,881 | 50,398 | 46,624 | 49,793 | 46,473 | 50,561 | 52,775 | 52,786 |
| NETHERLANDS | 465,196 | 483,999 | 504,300 | 499,980 | 387,699 | 482,531 | 555,812 | 502,454 | 417,036 | 387,553 | 449,350 | 382,825 | 414,306 | 443,531 |
| NORWAY | 109,907 | 109,164 | 129,195 | 110,617 | 98,675 | 127,754 | 138,345 | 137,967 | 142,151 | 144,202 | 150,686 | 154,603 | 158,650 | 147,929 |
| PORTUGAL | 206,488 | 194,702 | 201,816 | 213,389 | 161,013 | 223,464 | 153,404 | 95,309 | 105,921 | 142,826 | 178,503 | 207,345 | 222,129 | 228,327 |
| SPAIN | 1,528,877 | 1,634,608 | 1,614,835 | 1,161,176 | 952,772 | 982,015 | 808,051 | 699,589 | 722,689 | 890,125 | 1,094,077 | 1,147,007 | 1,234,932 | 1,321,438 |
| SWEDEN | 274,301 | 282,766 | 306,794 | 253,982 | 213,408 | 289,684 | 304,984 | 279,899 | 269,599 | 303,948 | 345,108 | 372,318 | 379,393 | 353,729 |
| SWITZERLAND (+FL) | 266,770 | 269,421 | 284,674 | 288,525 | 266,018 | 294,239 | 318,958 | 328,139 | 307,885 | 301,942 | 323,783 | 317,318 | 311,996 | 299,135 |
| UNITED KINGDOM | 2,439,717 | 2,344,864 | 2,404,007 | 2,131,795 | 1,994,999 | 2,030,846 | 1,941,253 | 2,044,609 | 2,264,737 | 2,476,435 | 2,633,503 | 2,692,786 | 2,540,617 | 2,367,147 |
| EUROPE NEW MEMBERS | 1,056,340 | 1,140,956 | 1,305,088 | 1,309,842 | 864,307 | 846,145 | 827,224 | 794,622 | 789,262 | 912,813 | 1,026,623 | 1,188,771 | 1,311,060 | 1,416,493 |
| BULGARIA* | 25,956 | 36,455 | 43,521 | 45,143 | 22,869 | 16,257 | 19,250 | 19,419 | 19,352 | 20,359 | 23,500 | 26,370 | 33,265 | 37,506 |
| CROATIA | 70,541 | 78,775 | 82,664 | 88,265 | 44,918 | 38,587 | 41,561 | 31,360 | 27,802 | 33,962 | 35,715 | 44,106 | 50,769 | 60,041 |
| CYPRUS | 17,687 | 18,639 | 22,878 | 22,241 | 14,981 | 14,088 | 13,480 | 10,123 | 7,102 | 8,276 | 10,344 | 12,643 | 13,127 | 13,135 |
| CZECH REPUBLIC | 151,699 | 156,686 | 174,456 | 182,554 | 167,708 | 169,580 | 173,595 | 174,009 | 164,736 | 192,314 | 230,857 | 259,693 | 271,595 | 261,437 |
| ESTONIA | 19,640 | 25,363 | 30,912 | 24,579 | 9,946 | 10,295 | 17,070 | 19,424 | 19,694 | 20,969 | 20,347 | 22,429 | 25,618 | 26,297 |
| HUNGARY | 198,982 | 187,676 | 171,661 | 153,278 | 60,189 | 43,476 | 45,094 | 53,059 | 56,139 | 67,476 | 77,171 | 96,552 | 116,265 | 136,601 |
| LATVIA | 10,467 | 14,234 | 21,606 | 22,217 | 7,515 | 7,970 | 13,234 | 10,665 | 10,636 | 12,452 | 13,765 | 16,359 | 16,698 | 16,878 |
| LITHUANIA | 16,602 | 25,582 | 32,771 | 19,831 | 5,367 | 6,365 | 10,980 | 12,165 | 12,163 | 14,503 | 17,085 | 20,320 | 25,836 | 32,382 |
| MALTA | 6,552 | 6,745 | 6,240 | 5,423 | 5,894 | 4,056 | 5,428 | 5,884 | 5,749 | 6,451 | 7,121 | 7,333 | 7,825 | 8,128 |
| POLAND | 207,007 | 224,728 | 277,427 | 319,190 | 276,220 | 315,855 | 277,427 | 272,719 | 289,913 | 327,709 | 354,975 | 416,123 | 486,352 | 531,889 |
| ROMANIA | 214,967 | 247,411 | 312,533 | 285,506 | 116,016 | 94,441 | 81,709 | 66,436 | 57,710 | 82,809 | 98,325 | 115,004 | 105,083 | 129,004 |
| SLOVAKIA | 56,916 | 59,084 | 59,700 | 70,040 | 74,717 | 64,033 | 68,203 | 69,268 | 65,998 | 72,237 | 77,968 | 88,165 | 96,105 | 98,080 |
| SLOVENIA | 59,324 | 59,578 | 68,719 | 71,575 | 57,967 | 61,142 | 60,193 | 50,091 | 52,268 | 53,296 | 59,450 | 63,674 | 62,522 | 65,115 |
| RUSSIA, TURKEY & OTHER EUROPE | 2,284,420 | 2,724,418 | 3,471,314 | 3,909,719 | 2,075,646 | 2,669,169 | 3,524,941 | 3,623,366 | 3,597,858 | 3,092,818 | 2,122,682 | 2,131,580 | 2,342,998 | 2,285,827 |

# Strategies for dealing with sub-headers

Steps:

1. Read in the data, skipping first 5 rows
2. Clean the names

```
pc_sales <- read_excel(
    here::here('data', 'pc_sales_2018.xlsx'),
    sheet = 'pc_sales', skip = 5) %>%
    clean_names() %>%
    rename(country = regions_countries)

glimpse(pc_sales)
```

```
#> Rows: 160
#> Columns: 18
#> $ country <chr> NA, "EUROPE", "EU 28 countries + E
#> $ x2      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA
#> $ x3      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA
#> $ x4      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA
#> $ x2005   <dbl> NA, 17906455, 15622035, 14565695,
#> $ x2006   <dbl> NA, 18685556, 15961138, 14820182,
#> $ x2007   <dbl> NA, 19618588, 16147274, 14842186,
#> $ x2008   <dbl> NA, 18821599, 14911880, 13602038,
#> $ x2009   <dbl> NA, 16608761, 14533115, 13668808,
#> $ x2010   <dbl> NA, 16499863, 13830694, 12984549,
#> $ x2011   <dbl> NA, 17167600, 13642659, 12815435,
```

# Strategies for dealing with sub-headers

Steps:

1. Read in the data, skipping first 5 rows
2. Clean the names
3. **Drop bad columns**
4. **Filter out bad rows**

Use **datapasta** to get rows to drop

```
drop <- c(
    'EUROPE', 'EU 28 countries + EFTA',
    'EU 15 countries + EFTA', 'EUROPE NEW MEMBERS',
    'RUSSIA, TURKEY & OTHER EUROPE', 'AMERICA',
    'NAFTA', 'CENTRAL & SOUTH AMERICA',
    'ASIA/OCEANIA/MIDDLE EAST', 'AFRICA', 'ALL COUNTRIES')

pc_sales <- pc_sales %>%
    select(-c(x2:x4)) %>%         # Drop bad columns
    filter(! country %in% drop, # Drop bad rows
           ! is.na(country))

head(pc_sales)
```

```
#> # A tibble: 6 × 15
#>    country     x2005    x2006    x2007    x2008    x2009    x2010
#>    <chr>       <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
#> 1 AUSTRIA    307915   308594   298182   293697   319403   328563
#> 2 BELGIUM    480088   526141   524795   535947   476194   547340
#> 3 DENMARK    148819   156936   162686   150199   112454   153858
#> 4 FINLAND    148161   145700   125608   139669    90574   111968
```

# Strategies for dealing with sub-headers

Steps:

1. Read in the data, skipping first 5 rows
2. Clean the names
3. Drop bad columns
4. Filter out bad rows
5. **Gather the year variables**

```
pc_sales <- pc_sales %>%
    gather(key = 'year', value = 'num_cars', x2005:x2018)

head(pc_sales)
```

```
#> # A tibble: 6 × 3
#>    country year   num_cars
#>    <chr>   <chr>      <dbl>
#> 1 AUSTRIA x2005     307915
#> 2 BELGIUM x2005     480088
#> 3 DENMARK x2005     148819
#> 4 FINLAND x2005     148161
#> 5 FRANCE  x2005    2118042
#> 6 GERMANY x2005    3319259
```

# Strategies for dealing with sub-headers

Steps:

1. Read in the data, skipping first 5 rows
2. Clean the names
3. Drop bad columns
4. Filter out bad rows
5. Gather the year variables
6. **Separate the "x" from the year**

```
pc_sales <- pc_sales %>%
    separate(year, into = c('drop', 'year'), sep = 'x',
             convert = TRUE)

head(pc_sales)
```

```
#> # A tibble: 6 × 4
#>   country drop   year num_cars
#>   <chr>   <lgl> <int>    <dbl>
#> 1 AUSTRIA NA     2005   307915
#> 2 BELGIUM NA     2005   480088
#> 3 DENMARK NA     2005   148819
#> 4 FINLAND NA     2005   148161
#> 5 FRANCE  NA     2005  2118042
#> 6 GERMANY NA     2005  3319259
```

# Strategies for dealing with sub-headers

Steps:

1. Read in the data, skipping first 5 rows
2. Clean the names
3. Drop bad columns
4. Filter out bad rows
5. Gather the year variables
6. Separate the "x" from the year
7. **Remove the drop column**
8. **Finish cleaning**

```r
pc_sales <- pc_sales %>%
  select(-drop) %>%
  mutate(country  = str_to_title(country))

head(pc_sales)
```

```
#> # A tibble: 6 × 3
#>   country  year num_cars
#>   <chr>   <int>    <dbl>
#> 1 Austria  2005   307915
#> 2 Belgium  2005   480088
#> 3 Denmark  2005   148819
#> 4 Finland  2005   148161
#> 5 France   2005  2118042
#> 6 Germany  2005  3319259
```

# What if I wanted to keep the continents?

Strategy: Join a new data frame linking country -> continent

```r
drop <- c(
  'EUROPE', 'EU 28 countries + EFTA',
  'EU 15 countries + EFTA', 'EUROPE NEW MEMBERS',
  'RUSSIA, TURKEY & OTHER EUROPE', 'AMERICA',
  'NAFTA', 'CENTRAL & SOUTH AMERICA',
  'ASIA/OCEANIA/MIDDLE EAST', 'AFRICA', 'ALL COUNTRIES')

pc_sales <- read_excel(
  here::here('data', 'pc_sales_2018.xlsx'),
  sheet = 'pc_sales', skip = 5) %>%
  clean_names() %>%
  rename(country = regions_countries) %>%
  select(-c(x2:x4)) %>%          # Drop bad columns
  filter(! country %in% drop,    # Drop bad rows
          ! is.na(country)) %>%
  gather(key = 'year', value = 'num_cars', x2005:x2018) %>%
  separate(year, into = c('drop', 'year'), sep = 'x',
           convert = TRUE) %>%
  select(-drop)

head(pc_sales, 3)
```

```
#> # A tibble: 3 × 3
#>   country  year num_cars
#>   <chr>    <int>   <dbl>
#> 1 AUSTRIA  2005   307915
#> 2 BELGIUM  2005   480088
#> 3 DENMARK  2005   148819
```

# Strategy 1: Find another source

# Strategy 2: Hand-make it

```
pc_regions <- read_csv(here::here(
  "data", "pc_regions.csv"))

head(pc_regions)
```

```
pc_sales <- pc_sales %>%
  left_join(pc_regions)

head(pc_sales)
```

```
#> # A tibble: 6 × 3
#>   country region subregion
#>   <chr>   <chr>  <chr>
#> 1 AUSTRIA EUROPE EU 15 countries + EFTA
#> 2 BELGIUM EUROPE EU 15 countries + EFTA
#> 3 DENMARK EUROPE EU 15 countries + EFTA
#> 4 FINLAND EUROPE EU 15 countries + EFTA
#> 5 FRANCE  EUROPE EU 15 countries + EFTA
#> 6 GERMANY EUROPE EU 15 countries + EFTA
```

```
#> # A tibble: 6 × 5
#>   country year num_cars region subregion
#>   <chr>   <int>    <dbl> <chr>  <chr>
#> 1 AUSTRIA 2005   307915 EUROPE EU 15 cou
#> 2 BELGIUM 2005   480088 EUROPE EU 15 cou
#> 3 DENMARK 2005   148819 EUROPE EU 15 cou
#> 4 FINLAND 2005   148161 EUROPE EU 15 cou
#> 5 FRANCE  2005  2118042 EUROPE EU 15 cou
#> 6 GERMANY 2005  3319259 EUROPE EU 15 cou
```

```r
drop <- c(
    'EUROPE', 'EU 28 countries + EFTA',
    'EU 15 countries + EFTA', 'EUROPE NEW MEMBERS',
    'RUSSIA, TURKEY & OTHER EUROPE', 'AMERICA',
    'NAFTA', 'CENTRAL & SOUTH AMERICA',
    'ASIA/OCEANIA/MIDDLE EAST', 'AFRICA', 'ALL COUNTRIES')

pc_regions <- read_csv(here::here("data", "pc_regions.csv"))

pc_sales <- read_excel(
    here::here('data', 'pc_sales_2018.xlsx'),
    sheet = 'pc_sales', skip = 5) %>%
    clean_names() %>%
    rename(country = regions_countries) %>%
    select(-c(x2:x4)) %>%          # Drop bad columns
    filter(! country %in% drop, # Drop bad rows
           ! is.na(country)) %>%
    gather(key = 'year', value = 'num_cars', x2005:x2018) %>%
    separate(year, into = c('drop', 'year'), sep = 'x',
             convert = TRUE) %>%
    select(-drop) %>%
    left_join(pc_regions) %>%
    mutate(
        country  = str_to_title(country),
        region  = str_to_title(region),
        subregion  = str_to_title(subregion))

head(pc_sales)
```

```
#> # A tibble: 6 × 5
#>   country   year num_cars region subregion
#>   <chr>    <int>    <dbl> <chr>  <chr>
#> 1 Austria   2005   307915 Europe Eu 15 Countries + Efta
#> 2 Belgium   2005   480088 Europe Eu 15 Countries + Efta
#> 3 Denmark   2005   148819 Europe Eu 15 Countries + Efta
#> 4 Finland   2005   148161 Europe Eu 15 Countries + Efta
#> 5 France    2005  2118042 Europe Eu 15 Countries + Efta
#> 6 Germany   2005  3319259 Europe Eu 15 Countries + Efta
```