

# Week 9: Cleaning Data

**並** EMSE 4575: Exploratory Data Analysis

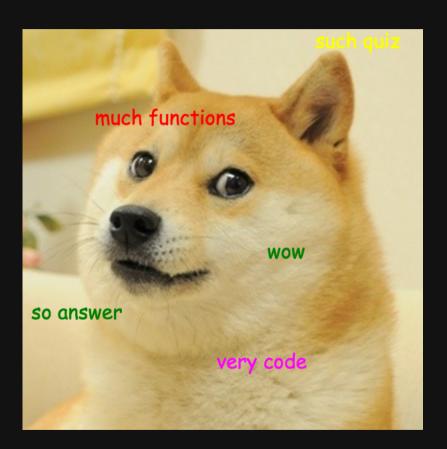
John Paul Helveston

**歯** March 10, 2021

# Quiz 3

Link is on the schedule

10:00



### **Waffles and Animations**

Waffles Animation

The bug: The bug:

install.packages("waffle")

animate(anim\_plot)

The fix:

The fix:

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

```
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>U.S. State Abbreviations</u>

### 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'))</pre>
```

### "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'))</pre>
```

### 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
- 6. Dealing with messy Excel files

## Week 9: Cleaning Data

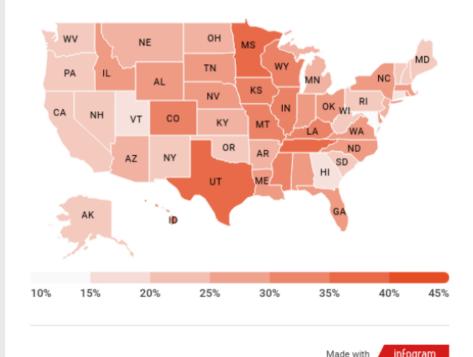
- 1. Merging datasets with joins
- 2. Are your variables the right type?
- 3. Are your variables the right *name*?

**BREAK** 

- 4. Re-coding variables
- 5. Dates
- 6. Dealing with messy Excel files

# A state breakdown of who's skipping medications because they're too costly

Across the U.S., 28% of consumers ages 19 to 64 say they have not taken their prescription drugs as their health care provider has prescribed them because of cost, according to AARP research. Here's a look at the percentage by state of residents who say they stopped taking medication due to cost.



## What's wrong with this map?

### Likely culprit: Merging two columns

```
head(names)

#> state_name
```

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

```
head(abbs)
```

```
result <- bind_cols(names, abbs)
head(result)</pre>
```

### Joins

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

Example: band\_members & band\_instruments

```
band_members
```

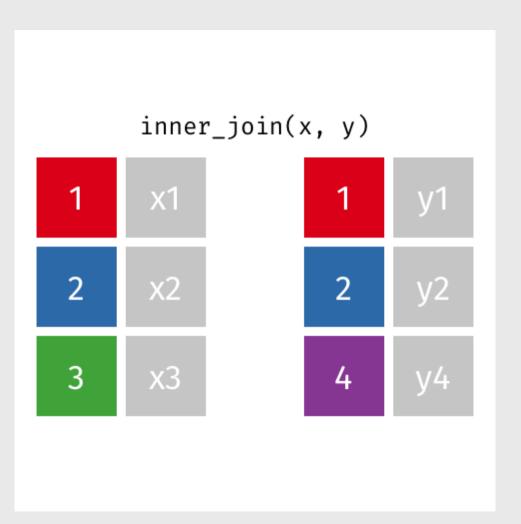
```
band_instruments
```

```
#> # 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)
```

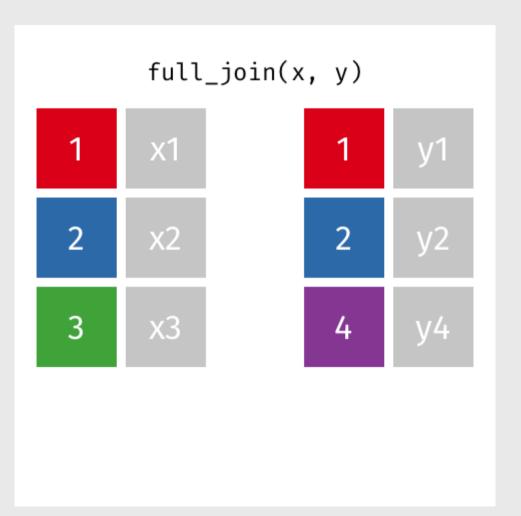
```
#> # A tibble: 2 x 3
#> name band plays
#> <chr> <chr> #> 1 John Beatles guitar
#> 2 Paul Beatles bass
```



## full\_join()

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

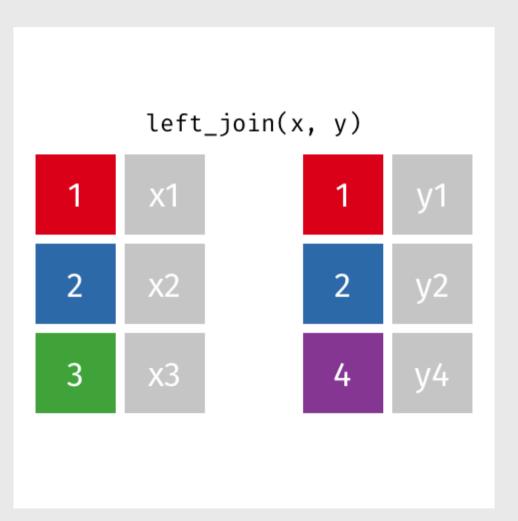
```
#> # A tibble: 4 x 3
#> name band plays
#> <chr> <chr> <chr>
#> 1 Mick Stones <NA>
#> 2 John Beatles guitar
#> 3 Paul Beatles bass
#> 4 Keith <NA> guitar
```



## left\_join()

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

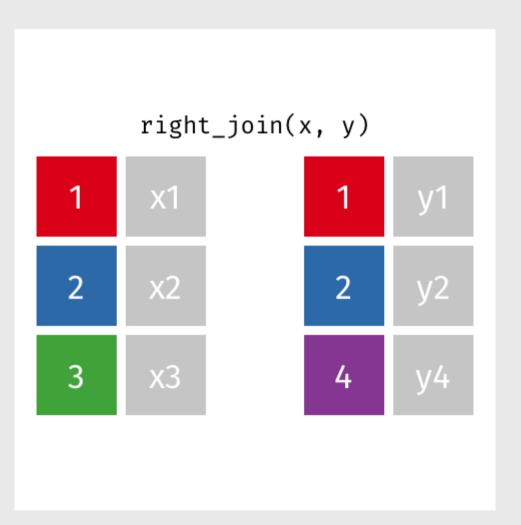
```
#> # A tibble: 3 x 3
#> name band plays
#> <chr> <chr> <chr>
#> 1 Mick Stones <NA>
#> 2 John Beatles guitar
#> 3 Paul Beatles bass
```



## 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> #> 1 Mick Stones <NA>
#> 2 John Beatles guitar
#> 3 Paul Beatles bass
```

```
#> # 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")

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

```
band_instruments2
```

```
#> # 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
```

### 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)

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
                                                                                                                                                                                        <chr> "Northeast", "Northeast", "Northeast", "Northeast"
 #> $ region
                                                                                                                                                                                       <chr> "Maine", "
 #> $ state name
                                                                                                                                                                                       <chr> "ME", "ME", "ME", "ME", "ME", "ME", "ME", "ME", "ME", "
 #> $ state abb
                                                                                                                                                                                       <dttm> 2018-10-23, 2018-10-07, 2018-10-05, 2018-10-05,
 #> $ incident date
                                                                                                                                                                                       <chr> "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "KPWM", "
 #> $ airport id
                                                                                                                                                                                       <chr> "PORTLAND INTL JETPORT (ME)", "PORTLAND INTL JETPORT", "PORTLAND 
 #> $ airport
 #> $ operator
                                                                                                                                                                                        <chr> "AMERICAN AIRLINES", "AMERICAN AIRLINES", "AMERICAN
                                                                                                                                                                                       <chr> "A-320", "A-319", "A-319", "EMB-190", "EMB-170",
 #> $ atvpe
                                                                                                                                                                                       #> $ type eng
                                                                                                                                                                                       <chr> "UNKBS", "ZX302", "ZS010", "I1102", "K3310", "YH00"
 #> $ species id
                                                                                                                                                                                       <chr> "Unknown bird - small", "Swamp sparrow", "Blackpo"
  #> $ species
                                                                                                                                                                                       <chr> "N", NA, "N", "M?", "N", "N", "N", "N", "N", "N",
 #> $ damage
                                                                                                                                                                                       #> $ num engs
 #> $ incident month
                                                                                                                                                                                       <dbl> 10, 10, 10, 7, 11, 11, 10, 7, 8, 11, 7, 5, 4,
                                                                                                                                                                                       <dbl> 2018, 2018, 2018, 2018, 2017, 2016, 2016, 2016, 2016
 #> $ incident year
                                                                                                                                                                                       <chr> NA, "Night", "Night", "Day", "Dawn", "Day", "Day"
 #> $ time of day
                                                                                                                                                                                        <dbl> 1310, 1035, 2200, 1645, 645, 1345, 1346, 1400, 11
                                                                                                                                                                                        <dbl> 15, NA, 1000, 0, 0, 0, 0, NA, NA, 2000, 0, 50, 0,
 #> $ heiaht
 #> $ speed
                                                                                                                                                                                        <dbl> 150, NA, 140, 110, NA, NA, NA, NA, NA, 250, 100,
                                                                                                                                                                                       <chr> "departure", "arrival", "arrival"
 #> $ phase of flt
                                                                                                                                                                                       <chr> "Overcast", "Some Cloud", "Some Cloud"
 #> $ sky
 #> $ precip
                                                                                                                                                                                       <chr> "None", "None", "None", "None", "None", "None", None", None", None", "None", "No
<ord> Tue, Sun, Fri, Fri, Tue, Mon, Mon, Sat, Sat, Wed,
 #> $ weekday name
```

# 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
- 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

```
wind <- read_excel(here::here(
   'data', 'US_State_Wind_Energy_Facts_2018.xlsx'))
glimpse(wind)</pre>
```

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

### 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</pre>
```

```
#> [1] "numeric" "text" "numeric" "numeric" "numer
wind <- read_excel(here::here(
   'data', 'US_State_Wind_Energy_Facts_2018.xlsx'),
   col_types = columns)
glimpse(wind)</pre>
```

```
#> Rows: 50
#> Columns: 7
#> $ Ranking
                                      <dbl> 1, 2, 3,
#> $ State
                                      <chr> "TEXAS",
#> $ `Installed Capacity (MW)`
                                      <dbl> 23262, 749
  $ `Equivalent Homes Powered`
                                      <dbl> 6235000,
  $ `Total Investment ($ Millions)`
                                      <dbl> 42000, 137
#> $ `Wind Projects Online`
                                      <dbl> 136, 45,
#> $ `# of Wind Turbines`
                                      <dbl> 12750, 373
```

```
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)</pre>
```

### readr::read\_csv()

col\_types describes individual variables by name using cols()

Туре	<pre>dplyr::glimpse()</pre>	readr::parse_*()	readr::col_*()
Logical	<lgl></lgl>	<pre>parse_logical()</pre>	col_logical()
Numeric	<int> or <dbl></dbl></int>	<pre>parse_number()</pre>	col_number()
Character	<chr></chr>	<pre>parse_character()</pre>	col_character()
Factor	<fct></fct>	parse_factor(levels)	col_factor(levels)
Date	<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.)
mutate(
   Ranking = as.numeric(Ranking),
   `Equivalent Homes Powered` = as.numeric(
   `Total Investment ($ Millions)` = as.numeric() # of Wind Turbines` = as.numeric() # of glimpse(wind)</pre>
```

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

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

# 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
- 6. Dealing with messy Excel files

```
janitor::clean_names()
```



```
wind <- read_excel(here::here(
   'data', 'US_State_Wind_Energy_Facts_2018.xlsx'))
glimpse(wind)</pre>
```

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

```
janitor::clean_names()
```



```
library(janitor)

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

glimpse(wind)
```

```
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)
```

```
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)
```

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

Example: data on sleeping patterns of different mammals

```
glimpse(msleep)
```

```
#> Rows: 83
#> Columns: 11
                 <chr> "Cheetah", "Owl monkey", "Mounta:
#> $ name
                 <chr> "Acinonyx", "Aotus", "Aplodontia
  $ genus
                 <chr> "carni", "omni", "herbi", "omni"
  $ vore
                 <chr> "Carnivora", "Primates", "Rodent:
  $ order
  $ conservation <chr>> "lc", NA, "nt", "lc", "domestica
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4
                <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4
  $ sleep rem
  $ sleep_cycle <dbl> NA, NA, NA, 0.1333333, 0.6666667
                 <dbl> 11.90, 7.00, 9.60, 9.10, 20.00,
#> $ awake
  $ 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()
```

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()
```

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

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

### Select columns based on their data type

### Select only numeric columns:

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

### Select only character columns:

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

### Use select() to reorder variables

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

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

```
#> Rows: 83
#> Columns: 11
                  <chr> "Cheetah", "Owl mo
#> $ name
                  <chr> "Acinonyx", "Aotus
#> $ genus
                  <chr> "carni", "omni", "
#> $ vore
                  <chr> "Carnivora", "Prim
#> $ order
  $ conservation <chr>> "lc", NA, "nt",
#> $ 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
                  <dbl> 50.000, 0.480, 1.3
  $ bodywt
```

```
#> Rows: 83
#> Columns: 11
#> $ conservation <chr>> "lc", NA, "nt", "lc", "domes
                  <dbl> 11.90, 7.00, 9.60, 9.10, 20.
#> $ awake
                  <chr> "Cheetah", "Owl monkey", "Md
#> $ name
                  <chr> "Acinonyx", "Aotus", "Aplodo
#> $ genus
                  <chr> "carni", "omni", "herbi",
#> $ vore
                  <chr> "Carnivora", "Primates", "Rd
#> $ order
                  <dbl> 12.1, 17.0, 14.4, 14.9, 4.0;
#> $ sleep total
#> $ sleep_rem
                  <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2
#> $ sleep_cycle
                  <dbl> NA, NA, NA, 0.1333333, 0.666
                  <dbl> NA, 0.01550, NA, 0.00029, 0.
#> $ brainwt
#> $ 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
                       <chr> "Acinonyx", "Aotus
#> $ genus
                       <chr> "carni", "omni",
  $ vore
                       <chr> "Carnivora", "Prim
#> $ order
#> $ extinction_threat <chr>> "lc", NA, "nt", "
                       <dbl> 12.1, 17.0, 14.4,
  $ sleep_total
#> $ 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,
                       <dbl> 50.000, 0.480, 1.3
  $ bodywt
```

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

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

### 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
                       <chr> "Acinonyx", "Aotus
#> $ genus
                       <chr> "carni", "omni",
  $ vore
                       <chr> "Carnivora", "Prim
#> $ order
  $ extinction_threat <chr>> "lc", NA, "nt", "]
  $ sleep_total
                       <dbl> 12.1, 17.0, 14.4,
#> $ sleep rem
                       <dbl> NA, 1.8, 2.4, 2.3
                       <dbl> NA, NA, NA, 0.133
  $ sleep_cycle
                       <dbl> 11.90, 7.00, 9.60,
  $ awake
                       <dbl> NA, 0.01550, NA,
  $ brainwt
                       <dbl> 50.000, 0.480, 1.3
#> $ bodywt
```

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", "
                       <chr> "Acinonyx", "Aotus
  $ genus
                       <chr> "carni", "omni",
  $ vore
                       <chr> "Carnivora", "Prim
#> $ order
  $ sleep total
                       <dbl> 12.1, 17.0, 14.4,
  $ sleep_rem
                       <dbl> NA, 1.8, 2.4, 2.3
                       <dbl> NA, NA, NA, 0.1333
  $ sleep_cycle
                       <dbl> 11.90, 7.00, 9.60,
  $ awake
```

### Your turn

Read in the <a href="hot\_dog\_winners.xlsx">hot\_dog\_winners.xlsx</a> file and adjust the variable names and types to the following:

15:00

$\mathbf{A}$	Α	В	С	D	E	F	G
1	Year	Mens	Dogs eaten	Country	Womens	Dogs eaten	Country
2	1980	Paul Siederman & Joe Baldini	9.1	<b>United States</b>			
3	1981	Thomas DeBerry	11	<b>United States</b>			
4	1982	Steven Abrams	11	<b>United States</b>			
5	1983	Luis Llamas	19.5	Mexico			
6	1984	Birgit Felden	9.5	Germany			
7	1985	Oscar Rodriguez	11.75	<b>United States</b>			
8	1986	Mark Heller	15.5	<b>United States</b>			
9	1987	Don Wolfman	12	<b>United States</b>			
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		Takeru Kobayashi	50*	Japan			
24	2002	Takeru Kobayashi	50.5*	Japan			
25		Takeru Kobayashi	44.5	Japan			
26		Takeru Kobayashi	53.5*	Japan			
27	2005	Takeru Kobayashi	49	Japan			
28		Takeru Kobayashi	53.75*	Japan			
29		Joey Chestnut	66*	United States			
30		Joey Chestnut	59	United States			
31		Joey Chestnut	68*	United States			
32		Joey Chestnut	54	United States			
33		Joey Chestnut	62	United States	Sonya Thomas	40*	United States
34		Joey Chestnut	68		Sonya Thomas		United States
35	_	Joey Chestnut	69*		Sonya Thomas		United States
36		Joey Chestnut	61	United States		34	United States
37	_	Matt Stonie	62	United States		38	United States
38		Joey Chestnut	70*	United States		38.5	United States
39		Joey Chestnut	72*	United States		41	United States
40		Joey Chestnut	74*	United States		37	United States
41		Joey Chestnut	71	United States		31	United States
42	2019	Joey Chestriat	/1	Officed States	IVIIKI JUUU	31	Onited States
	Natar	• manne navy record					
43	Notes:	* means new record					

# Break!

Stand up, Move around, Stretch!



# 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
- 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.

```
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> <lql>
     1000
                26783 FALSE
    200
                26783 FALSE
    10000
                26783 FALSE
  4 100000
                26783 TRUE
    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()
```

# 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()
```

# 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()
```

# 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
               vear 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
                             population
#>
    country year cases
    <chr>
                <int> <chr> <chr>
                            19987071
#> 1 Afghanistan 1999 745
#> 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
               vear 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
```

```
#> # A tibble: 6 x 4
                             population
    country
               year cases
    <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 4
#>
    country year
                      cases population
           <int> <int>
#>
    <chr>
                                <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
               vear 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)
```

```
\#>\# A tibble: 6\times4
#>
     country
                 century year
                               rate
    <chr>
                         <chr> <chr>
                 <chr>
#> 1 Afghanistan 19
                         99
                               745/19987071
#> 2 Afghanistan 20
                         00
                               2666/20595360
#> 3 Brazil
                               37737/172006362
#> 4 Brazil
                 20
                         00
                               80488/174504898
                               212258/1272915272
#> 5 China
                 19
                         99
#> 6 China
                 20
                         00
                               213766/1280428583
```

# unite(): The opposite of separate()

```
tb_rates
```

```
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
```

```
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
```

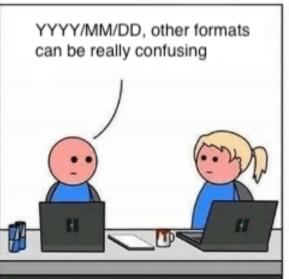
# Week 9: Cleaning Data

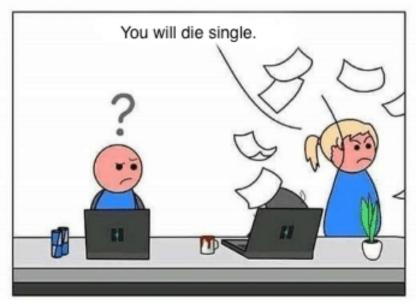
- 1. Merging datasets with joins
- 2. Are your variables the right type?
- 3. Are your variables the right *name*?

**BREAK** 

- 4. Re-coding variables
- 5. Dates
- 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')		

"2020-02-26"

# Check out the lubridate cheat sheet

# Extracting information from dates

```
date <- today()
date

#> [1] "2021-03-10"

# Get the year
year(date)

#> [1] 2021
```

# Extracting information from dates

```
date <- today()</pre>
date
#> [1] "2021-03-10"
# Get the year
                                                             # Get the day
year(date)
                                                             day(date)
#> [1] 2021
                                                             #> [1] 10
# Get the month
                                                             # Get the weekday
month(date)
                                                             wday(date)
#> [1] 3
                                                             #> [1] 4
# Get the month name
                                                             # Get the weekday name
month(date, label = TRUE, abbr = FALSE)
                                                             wday(date, label = TRUE, abbr = TRUE)
#> [1] March
                                                             #> [1] Wed
#> Levels: January < February < March < April < May < J</pre>
                                                             #> Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat</pre>
```

# 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</pre>
```

# Modifying date elements

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

```
date <- today()</pre>
date
#> [1] "2021-03-10"
# Change the year
year(date) <- 2016
date
#> [1] "2016-03-10"
# Change the day
day(date) <- 30
date
```

# Quick practice

What do you think will happen if we do this?

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

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 'takeoff run' should be merged into a single value called 'departure'.
  - All other values should be called 'other'.

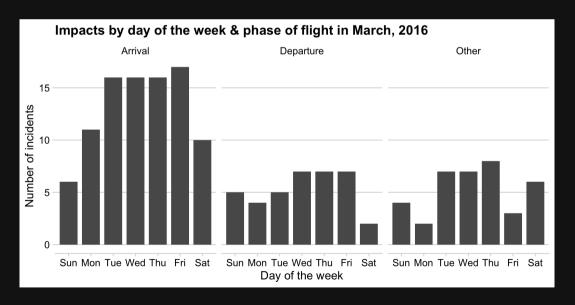
#### Before:

unique(str\_to\_lower(wildlife\_impacts\$phase\_of\_flt))
#> [1] "climb" "landing roll" NA "approximation"

#### 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.



# 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
- 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

7	Α	В	С	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		Luis Llamas	19.5	Mexico			
6	1984	Birgit Felden	9.5	Germany			
7	1985	Oscar Rodriguez	11.75	United States			
8		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	_	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		Matt Stonie	62	United States	Miki Sudo	38	United States
38	2016	Joey Chestnut	70*	United States	Miki Sudo	38.5	United States
39		Joey Chestnut	72*	United States	Miki Sudo	41	United States
40		Joey Chestnut	74*	United States		37	United States
41		•	71	United States	Miki Sudo	31	United States
42		•					
43	Notes	: * means new record					
38 39 40 41 42	2016 2017 2018 2019	Joey Chestnut Joey Chestnut Joey Chestnut Joey Chestnut	70* 72* 74*	United States United States United States	Miki Sudo Miki Sudo Miki Sudo	38.5 41 37	United Stat United Stat United Stat

# Strategy 1: divide & conquer

### 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)
```

# 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

```
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)
```

		В	С	D	E	F	G
1	Year	Mens	Dogs eaten	Country	Womens	Dogs eaten	Country
2	1980	Paul Siederman & Joe Baldini	9.1	<b>United States</b>			
3	1981	Thomas DeBerry	11	<b>United States</b>			
4	1982	Steven Abrams	11	<b>United States</b>			
5	1983	Luis Llamas	19.5	Mexico			
6	1984	Birgit Felden	9.5	Germany			
7	1985	Oscar Rodriguez	11.75	<b>United States</b>			
8	1986	Mark Heller	15.5	<b>United States</b>			
9	1987	Don Wolfman	12	<b>United States</b>			
10	1988	Jay Green	14	<b>United States</b>			
11	1989	Jay Green	13	<b>United States</b>			
12	1990	Mike DeVito	16	<b>United States</b>			
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	<b>United States</b>	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 x 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
#> 6 1985 Oscar Rodriguez
                                                 9.5 Germany Mens
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)
```

# 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

```
#> # A tibble: 3 x 3
#> year variable value
#> <chr> <chr> 
#> 1 1980 competitor.mens Paul Siederman & Joe Baldi
#> 2 1981 competitor.mens Thomas DeBerry
#> 3 1982 competitor.mens Steven Abrams
72 /
```

## Strategy 2: gather, separate, spread

- 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

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

## Strategy 2: gather, separate, spread

- 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)
```

#### Divide & conquer

```
hot dogs <- read excel(</pre>
    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(
        vear.
        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

```
hot dogs <- read excel(</pre>
   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', 'competition')
            sep = '\\.') %>%
   # Spread "joint" variables
   spread(key = variable, value = value) %>%
   # Finish cleaning
   mutate(
       new record = str detect(dogs eaten, "\\*"),
        dogs eaten = parse number(dogs eaten),
                  = as.numeric(year))
       year
```

Example:

OICA passenger car sales data

7	Α	E	F	G	н	1	J	К	L	М	N	0	Р	Q	R
1 2	<b>@</b> OICA		NEW PO	REGIS	TRATIC	NS OR	SALES								
3	WOICH														
5		Estimated fig	ures												
6	REGIONS/COUNTRIES	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
7															
8	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
9	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
10	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
11	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
12	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
13	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
14	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
15	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
16	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
17	GREECE ICELAND	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
18	IRELAND	18,060 171,742	17,129 178,484	15,942 186.325	9,033 151,607	2,113 57.453	3,106 88,446	5,038 89,911	7,902 79,498	7,274 74,367	9,537 96,284	14,004 124,804	18,442 146.600	21,324 131,332	17,976 125,557
19	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
21	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
22	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
23	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
24	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
25	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
26	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
27	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
28	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
29	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
30	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
31	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
32	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
33	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
34	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
35	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
36	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
37	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
38	MALTA POLAND	6,552	6,745	6,240	5,423	5,894	4,056	5,428	5,884	5,749	6,451 327,709	7,121	7,333	7,825	8,128
39 40	ROMANIA	207,007 214,967	224,728 247,411	277,427 312,533	319,190 285,506	276,220 116,016	315,855 94,441	277,427 81,709	272,719 66,436	289,913 57,710	82,809	354,975 98,325	416,123 115,004	486,352 105,083	531,889 129,004
41	SLOVAKIA	56,916	59,084	59,700	70,040	74,717	64,033	68,203	69,268	65,998	72,237	98,325 77.968	88,165	96,105	98,080
42	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
43	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
43	NOOSIA, TURNET & OTHER EUROPE	2,204,420	2,724,410	3,471,314	3,303,119	2,070,040	2,009,109	3,324,341	3,023,300	3,391,656	3,002,010	2,122,002	2,131,380	2,342,390	2,200,021

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

```
pc sales <- read excel(</pre>
    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 + B
#> $ x2
            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
#> $ x3
#> $ x4
            <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
           <dbl> NA, 17906455, 15622035, 14565695,
#> $ x2005
#> $ x2006
            <dbl> NA, 18685556, 15961138, 14820182,
  $ x2007
            <dbl> NA, 19618588, 16147274, 14842186,
            <dbl> NA, 18821599, 14911880, 13602038,
#> $ x2008
            <dbl> NA, 16608761, 14533115, 13668808,
  $ x2009
            <dbl> NA, 16499863, 13830694, 12984549, 85
  $ x2010
#> $ x2011
            <dbl> NA, 17167600, 13642659, 12815435,
```

#### 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 x 15
    country x2005
                      x2006
                              ×2007
                                      ×2008
                                              x2009
                                                       x2010
    <chr>
              <dbl>
                      <dbl>
                              <dbl>
                                      <dbl>
                                               <dbl>
                                                       <dbl>
                             298182
  1 AUSTRIA 307915
                      308594
                                     293697
                                             319403
                                                      328563
  2 BELGIUM
             480088
                      526141
                             524795
                                     535947
                                              476194
                                                      547340
                                      150199
    DENMARK
              148819
                      156936
                              162686
                                              112454
                                                      153858
                                               90574
                                                      111968
#> 4 FINLAND
             148161
                     145700
                             125608
                                      139669
```

- 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 x 3
    country year
                  num cars
    <chr> <chr>
                     <dbl>
#> 1 AUSTRIA x2005
                   307915
#> 2 BELGIUM x2005
                    480088
    DENMARK ×2005
                   148819
  4 FINLAND x2005
                   148161
  5 FRANCE ×2005
                   2118042
  6 GERMANY ×2005
                   3319259
```

- 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

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

- 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

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

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

## What if I wanted to keep the continents?

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

```
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(</pre>
 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 x 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)</pre>
```

```
#> # A tibble: 6 x 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
```

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

```
#> # A tibble: 6 x 5
     country year num cars region subregion
    <chr>
             <int>
                      <dbl> <chr> <chr>
  1 AUSTRIA
              2005
                     307915 EUROPE EU 15
  2 BELGIUM
                     480088 EUROPE EU 15 cou
              2005
    DENMARK
                     148819 EUROPE EU 15 cou
              2005
    FINLAND
              2005
                     148161 EUROPE EU 15
              2005
  5 FRANCE
                    2118042 FUROPE FU 15 cou
  6 GERMANY
              2005
                    3319259 EUROPE EU 15 cou
```

REGIONS/COUNTRIES	Q R	o	P	0	N	м	L	K	1	1	н	G	F	E	A	$\mathbf{Z}$
FREGIONS/COUNTRIES   2005   2006   2007   2008   2009   2010   2011   2012   2013   2014   2015   2016									SALES	NS OR	TRATIC	C REGIS			@OICA	3 4
Fig. 2015   Fig.																_
BUILDINGE 17, 286,445 18,885,586 19,815,386 18,821,590 16,869,781	2017 2018	20	2016	2015	2014	2013	2012	2011	2010	2009	2008	2007	2006	2005	REGIONS/COUNTRIES	
10   EU 28 countries + EFTA   15,822,035   15,961,338   15,147,274   14,911,880   14,533,115   13,800,994   33,442,690   12,567,003   22,444,415   13,061,645   14,287,881   15,160,209   10   10   10   10   10   10   10			47 004 040	40 440 500	40 454 070	45.040.070	40 404 000	47 467 666	40 400 000	40 000 704	40 004 500	40.040.500	40.005.550	47 000 455	FURARE	
10 EU 15 countries + EFTA	17,974,281 17,912,33							,,		,,			, ,			-
11 AUSTRA 307915 306,594 228,192 233,697 319,403 328,863 356,145 336,010 319,035 303,316 308,555 326,604 13   12 BERGIUM 480,086 526,141 524,795 535,947 476,194 547,340 572,211 486,737 486,085 482,939 501,086 538,519 1   13 DENAMRK 148,819 156,939 162,686 150,199 112,464 153,868 170,036 170,783 182,086 182,085 207,717 222,024   14 FINLAND 144,161 145,700 125,608 139,669 90,574 111,988 126,123 111,251 103,465 106,237 108,819 119,917 1   15 FRANCE 2,118,042 2,045,745 2,108,672 2,091,369 2,023,388 2,251,604 2,044,229 1,868,760 1,796,466 1,796,865 1,917,226 2,015,177 1   16 GERMANY 3,19,299 3,467,981 3,146,163 3,009,040 3,907,175 2,916,259 3,178,564 3,082,504 2,982,431 3,089,773 3,206,042 3,351,607 1   17 GRECE 269,728 267,669 279,745 267,295 219,730 141,097 97,680 58,42 58,642 58,647 1,1218 7   18 IDELAND 110,000 17,129 15,942 9,033 2,113 3,106 5,038 7,902 7,274 9,537 14,004 18,442 1   18 IRELAND 177,742 178,448 168,355 151,607 57,433 84,68 8,991 7,902 7,274 9,537 14,004 18,442 1   1 LUXEMBOURG 48,517 50,337 51,332 52,599 482,51 1,961,580 1,748,740 1,403,010 1,304,648 1,369,576 1,575,737 1,824,968 1   1 LUXEMBOURG 48,517 50,337 51,332 52,599 482,231 555,812 50,2454 417,036 337,653 3449,339 382,825 1   2 NETHERLAND 109,007 109,164 128,165 110,817 98,675 127,754 183,404 98,309 105,921 142,826 178,030 382,825 1   2 NETHERLAND 200,488 194,702 201,816 213,380 161,013 224,44 153,404 98,309 105,921 142,226 178,030 382,825 1   2 NETHERLAND 200,488 194,702 201,816 213,380 161,013 224,44 153,404 98,309 105,921 142,226 178,030 382,825 1   2 NETHERLAND 109,907 109,164 128,165 110,877 98,275 982,015 980,016 199,989 309,984 344,933 382,825 1   2 NETHERLAND 200,488 194,702 201,816 213,380 161,013 224,464 153,404 98,309 105,921 142,226 178,030 382,825 1   2 NETHERLAND 178,707 198,942 149,999 200,998 309,984 344,933 307,885 319,992 309,985 331,918 31,9	15,631,283 15,626,50 14,320,223 14,210,01	_											.,,			-
12 BLGIUM 48,088 526,141 S24,785 535,947 476,194 577,241 486,073 486,065 482,039 501,066 539,519 10 DEMARK 1448,191 156,369 630 162,686 150,199 112,454 153,368 170,036 170,036 170,783 182,066 189,055 207,717 222,041 111,069 170,036 170,03	353.320 341.06															
13 DEMARK 148,819 156,395 126,868 150,199 112,454 153,869 170,036 170,783 182,066 180,055 207,777 222,924 15 FRANCE 2,118,042 2,046,745 2,106,872 2,091,389 2,302,388 2,281,689 2,244,229 1,898,760 1,790,456 1,795,885 1,917,226 2,015,177 16 GERMANY 3,192,299 3,467,861 3,146,163 3,090,040 3,807,175 2,916,259 1,926,269 1,926,241 3,096,773 3,206,042 3,351,607 17 GRECE 289,728 287,069 279,745 267,295 219,730 141,05 9,760 58,462 8,642 8,644 17,218 1,917,226 2,015,177 18 IECELAND 15,000 17,142 178,449 186,325 151,807 57,433 88,461 89,911 79,468 17,247 41,367 96,284 11,128 11,12	546,558 549,63															
148,161	221,821 218,56												,	,		
15 FARACE	120.480 120.48															
Semanny   3,319,299   3,467,891   3,146,163   3,000,040   3,807,175   2,916,299   3,173,634   3,082,904   2,962,431   3,008,775   3,206,042   3,351,607   3,206,042   3,206,	2.110.748 2.173.48															
The color of the	3,441,262 3,435,77															
18   ISCLAND   16,000   17,120   15,942   9,033   2,113   3,106   5,038   7,902   7,274   9,537   14,004   18,442   18,325   15,807   57,453   88,446   88,325   15,807   57,453   88,446   88,325   17,402   14,002   17,402   17	88,083 103,43															
19   INELAND   171,742   178,484   188,325   151,007   57,453   88,446   89,911   79,488   74,387   98,284   124,804   146,800	21,324 17,97															
20   TALY   2,244,108   2,335,462   2,491,115   2,181,390   2,194,865   1,491,590   1,749,740   1,403,010   1,304,648   1,380,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,575,737   1,824,968   1,280,576   1,275,746   1,285,477   1,284,968   1,280,576   1,275,476   1,280,576   1,275,476   1,280,476   1,280,576   1,275,476   1,280,476   1,28	131,332 125,55															
22 NORWY 109 907 109.164 129.195 110.817 98.00 387.699 482.531 555.812 502.454 417.036 387.553 449.350 382.825 28 NORWY 109.907 109.164 129.195 110.817 98.075 127.318 3.455 137.967 142.151 144.202 109.686 115.086 1	1,970,497 1,910,02	8 1,97	1,824,968	1,575,737	1,360,578			1,749,740	1,961,580	2,159,465		2,494,115	2,335,462	2,244,108	ITALY	20
28 NORWAY 109,907 109,164 128,195 110,877 88,675 127,7764 138,345 137,967 142,151 144,202 150,686 154,603 24 PORTUGAL 20,648 194,702 201,816 213,389 161,013 223,464 153,404 95,309 105,921 144,286 178,503 207,345 55 SPAIN 1,528,877 1,634,001,816,195 1,611,776 92,772 98,015 800,015 169,890 722,689 90,125 178,503 207,345 70,258 800,015 10,000 10,	52,775 52,78	31 5	50,561	46,473	49,793	46,624	50,398	49,881	49,726	47,265	52,359	51,332	50,837	48,517	LUXEMBOURG	21
24         PORTUGAL         206,488         194,702         201,816         213,389         161,013         223,464         153,404         95,309         105,921         142,262         178,503         207,345           25         SPAN         1,528,877         1,534,609         1,61,178         92,772         982,772         982,098         296,969         292,969         303,048         34,108         372,318         229,899         296,969         303,048         341,00         372,318         278,979         299,999         296,969         303,048         341,00         372,318         273,318         289,684         344,984         279,899         299,999         303,948         341,00         372,318         273,318         289,684         344,984         279,899         299,999         303,948         341,00         372,318         273,318         294,674         288,525         266,018         294,239         318,958         325,139         307,885         307,985         332,783         317,318         29,478         294,674         288,525         266,018         294,239         318,958         325,139         307,885         307,985         332,783         337,318         22,478,445         288,262         266,018         484,487         294,142,22	414,306 443,53	15 41	382,825	449,350	387,553	417,036	502,454	555,812	482,531	387,699	499,980	504,300	483,999	465,196	NETHERLANDS	22
25         SANN         1,528,877         1,634,809         1,164,835         1,161,789         962,772         808,051         699,889         722,689         890,125         1,094,077         1,472,072           25         SWEDEN         274,301         222,766         300,748         253,882         221,408         289,849         304,984         229,999         296,950         303,948         345,108         372,378           27         SWITZER,AND (+L)         266,770         299,421         224,674         288,825         266,018         294,239         318,988         328,139         307,885         301,942         323,853         317,318           28         EUROPE NEW MEMBERS         1,585,940         1,140,956         1,350,9842         348,445         22,869,778         1,942,999         2,003,846         1,941,293         2,244,737         2,476,435         2,833,503         2,692,786           29         EUROPE NEW MEMBERS         1,585,940         1,140,956         1,350,9842         484,145         22,869         19,257         19,250         19,419         19,352         20,359         233,503         23,662,786           31         CRECH REPUBLIC         151,699         166,688         1,364,749         22,269         16,257 </th <td>158,650 147,92</td> <td>3 15</td> <td>154,603</td> <td>150,686</td> <td>144,202</td> <td>142,151</td> <td>137,967</td> <td>138,345</td> <td>127,754</td> <td>98,675</td> <td>110,617</td> <td>129,195</td> <td>109,164</td> <td>109,907</td> <td>NORWAY</td> <td>23</td>	158,650 147,92	3 15	154,603	150,686	144,202	142,151	137,967	138,345	127,754	98,675	110,617	129,195	109,164	109,907	NORWAY	23
22 SWEEN 274.301 282.766 306,704 253.862 213.408 289.884 304.984 279.899 269.590 303.946 345,108 372.318 27 SWITZERLAND (+FL) 266,770 269,421 284,674 288,525 266,018 294.239 818,958 328,139 307,885 301,942 323,783 317,318 28 LINTED KINDDOM 2,439,777 2,344,564 2,404,007 2,131,795 1,994,999 2,039,46 1,941,253 2,044,509 2,2264,737 2,476,435 2,633,503 2,062,786 29 EUROPE NEW MEMBERS 1,069,340 1,140,958 1,305,088 1,309,42 848,307 846,145 827,224 794,622 789,262 19,128,13 1,026,623 1,189,771 31 CROATINA 70,541 78,775 82,664 88,265 44,918 38,587 41,561 31,360 27,802 33,962 35,715 44,106 22 CYPRUS 17,687 18,689 22,878 22,241 44,981 14,981 14,981 14,081 13,480 11,023 7,102 8,276 10,344 12,043 13 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 14,140,140 14,	222,129 228,32	5 22	207,345	178,503	142,826	105,921	95,309	153,404	223,464	161,013	213,389	201,816	194,702	206,488	PORTUGAL	24
27 SWITZER,AND (+FL) 266,770 299,421 284,674 288,525 286,018 294,239 318,988 328,139 307,885 301,942 323,783 317,318 217,318 218,019 2	1,234,932 1,321,43	7 1,23	1,147,007	1,094,077	890,125	722,689	699,589	808,051	982,015	952,772	1,161,176	1,614,835	1,634,608	1,528,877	SPAIN	25
28 UNTEO KINGDOM 2,439.717 2,344.864 2,406.007 2,131.705 1,994.990 2,030.846 1,941.263 2,044.900 2,264.737 2,476.435 2,633.503 2,692.786 9 EUROPE NEW MEMBERS 1,056,340 1,140,956 1,356,888 1,309,842 864,307 846,145 57 19,250 194,419 19,352 20,359 126,623 1,188,771 30 BULGARIA* 22,896 38,455 43,521 45,143 22,899 18,257 19,250 194,419 19,352 20,359 26,370 31 CROATINA 70,541 78,775 82,664 88,265 44,918 39,887 41,561 31,360 27,802 33,962 35,715 44,106 22 CVPRLS 17,687 115,689 2,2878 22,241 14,981 14,981 14,981 14,081 13,480 11,0123 7,102 8,276 10,344 12,043 13 CZECH REPUBLIC 1515,699 156,886 174,456 182,554 167,708 169,580 173,595 174,009 164,736 192,314 230,857 22,437 18,140 11,045 1	379,393 353,72	8 37	372,318	345,108	303,948	269,599	279,899	304,984	289,684	213,408	253,982	306,794	282,766	274,301	SWEDEN	26
29 EUROPE NEW MEMBERS 1,085,340 1,140,956 1,305,048 1,309,422 864,307 846,145 827,224 794,622 789,262 91,2613 1,026,623 1,188,771 30 BULGARIA* 25,956 36,455 45,143 22,669 16,257 19,250 19,419 19,352 20,359 23,500 26,370 1 CROATIA 70,541 79,775 82,664 88,265 44,918 38,277 41,561 31,300 27,602 33,962 33,962 35,715 44,106 32 CPFRUS 17,667 16,539 22,876 22,271 14,961 14,068 13,460 10,123 7,102 8,276 10,344 12,643 31 CECH REPUBLIC 151,669 156,666 174,456 182,554 167,708 169,577 17,09 164,736 192,314 220,857 220,857 33,912 14,913 14,914 14,915 18,371 14,915 14,709 164,736 192,314 220,857 220,857 33,912 14,914 14,915 18,371 14,915 14,709 164,736 192,314 220,857 220,857 33,912 14,914 14,915 14,	311,996 299,13															27
30 BULGARIA* 25,996 36,455 43,521 45,143 22,869 16,257 19,250 19,419 19,352 20,359 23,500 26,370 31 CROATIA 70,541 78,775 82,664 88,265 44,918 38,587 41,561 31,360 12,7802 39,962 35,715 44,106 32 CYPRUS 17,687 16,369 22,878 22,241 44,981 14,981 13,480 10,123 7,102 8,276 10,344 12,061 33 CZECH REPUBLIC 151,689 156,686 174,456 182,554 167,708 169,580 173,595 174,009 164,736 192,314 230,857 259,683 48 ESTOMA 19,640 25,363 30,912 24,579 9,946 10,258 17,070 19,424 19,649 42 9,969 20,347 22,4579 9,946 10,258 17,070 19,424 19,649 42 9,969 20,347 22,4579 9,946 10,258 17,070 19,424 19,649 42 9,969 20,347 22,4579 9,946 10,258 17,070 19,441 19,449 19,464 20,969 30,477 22,578 14,078 1	2,540,617 2,367,14	6 2,54	2,692,786	2,633,503	2,476,435	2,264,737	2,044,609	1,941,253	2,030,846	1,994,999	2,131,795	2,404,007	2,344,864	2,439,717	UNITED KINGDOM	28
31         CROATIA         70,541         78,775         82,664         88,265         44,918         38,887         41,561         31,360         27,802         33,962         35,715         44,106           32         CYPRUS         17,667         18,639         22,287         41,981         14,089         13,480         10,232         7,102         8,276         10,344         12,224           33         CZECH REPUBLIC         151,699         156,686         174,406         162,554         167,708         169,580         173,095         174,009         164,736         192,314         230,857         256,693           34         ESTONA         19,640         25,383         30,912         24,579         9,946         10,256         170,701         14,424         19,694         20,969         20,347         22,498           35         HUNGARY         198,982         187,676         171,691         153,279         9,046         10,256         170,701         14,244         19,694         20,969         20,347         27,479           36         LATVIA         10,467         14,234         21,606         22,217         7,515         7,970         13,234         10,665         10,636         17,717 <th>1,311,060 1,416,49</th> <th></th> <th></th> <th>.,,</th> <th></th> <th>,</th> <th>,</th> <th></th> <th></th> <th></th> <th>.,,</th> <th>.,,</th> <th></th> <th>.,,</th> <th></th> <th>29</th>	1,311,060 1,416,49			.,,		,	,				.,,	.,,		.,,		29
32 CZECH REPUBLIC 151,689 156,686 22,878 22,241 14,981 14,088 13,480 10,123 7,102 8,276 10,344 12,643 33 CZECH REPUBLIC 151,689 156,686 174,456 182,554 167,708 169,580 173,595 174,009 164,736 192,314 230,857 259,693 15 HUNGARY 198,682 187,676 171,681 153,278 60,189 43,476 45,094 50,599 66,139 67,476 77,171 96,552 17 HUNGARY 10,467 14,234 21,666 122,177 7,515 7,970 13,234 10,665 10,636 12,455 16,359 17,085 17,085 17,095 12,165 12,163 12,145 17,085 12,327 12,141,141,141,141,141,141,141,141,141,1	33,265 37,50															30
33 CECH REPUBLIC 151,699 166,886 174,486 182,554 167,708 169,890 173,995 174,009 164,736 192,314 230,857 256,893 18 ESTONIA 19,840 25,983 30,912 24,579 9,946 10,295 177,070 19,424 19,894 20,989 20,347 22,429 15 HUNGARY 186,892 187,676 171,661 153,278 60,189 43,76 45,094 53,099 56,139 67,476 77,171 98,552 16 LATVIA 10,487 14,224 21,605 22,217 7,515 7,970 13,234 10,685 10,686 17,478 11,484 11,	50,769 60,04															
34 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 35 HUNGARY 198,982 187,676 171,661 153,278 60,189 43,476 45,094 50,095 66,139 67,476 77,171 96,552 36 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 37 LITHAINIA 16,602 25,582 32,771 19,831 5,367 6,367 6,085 10,980 12,165 12,163 14,503 17,085 20,320	13,127 13,13															
35 HUNGARY 198.982 187,676 171,661 153,278 60,189 43,476 45,094 53,059 56,139 67,476 77,171 96,582 36 LATVIA 10,467 14,224 21,606 22,217 7,515 7,970 13,234 10,665 10,636 12,482 13,765 16,389 17,114,114,114,114,114,114,114,114,114,1	271,595 261,43															
36 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 37 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	25,618 26,29															
37 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	116,265 136,60															
	16,698 16,87															
	25,836 32,38 7,825 8,12															
39 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 40 ROMANIA 214,967 247,411 312,533 285,060 118,016 44,411 81,709 64,348 57,710 82,809 83,25 115,004	486,352 531,88 105,083 129,00															
40 ROMANIA 214,907 247,911 312,933 269,506 116,010 94,441 61,709 69,430 77,710 62,609 99,525 115,009 115,010 1	96.105 98.08															
42 SLOVENIA 59,324 59,578 68,719 71,579 75,7967 61,142 60,193 50,203 69,208 53,296 59,406 63,659	62.522 65.11						,	,								
42 SLUSSIA TURKEY & OTHER EUROPE 2,284,420 2,724,418 3,471,314 3,909,719 2,075,546 2,659,159 3,524,941 3,623,366 3,597,859 3,092,818 2,122,659 2,131,550	2,342,998 2,285,82	_			_											

```
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"))</pre>
pc sales <- read excel(</pre>
  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 x 5
#> country year num_cars region subregion
#> <chr> <int> <dbl> <chr> <chr> <int> 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
```