



Week 9: *Data Wrangling*

🏛️ EMSE 4571 / 6571: Intro to Programming for Analytics

👤 John Paul Helveston

📅 March 28, 2024

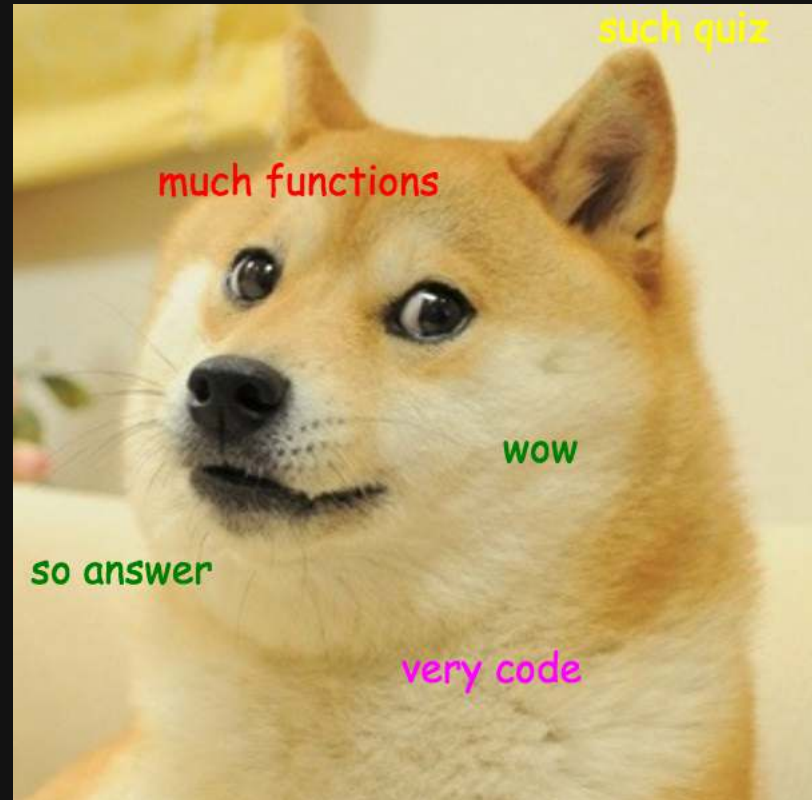
Quiz 5

10:00

Write your name on the quiz!

Rules:

- Work alone; no outside help of any kind is allowed.
- No calculators, no notes, no books, no computers, no phones.



How to fail with ChatGPT

1. Copy-paste the code and never run it.

```
(df$Population / df$State.Population) * 100
```

1. Do things I've explicitly told you not to do.

```
df <- read.csv("data/prisoners2019.csv") # Bad
```

```
df <- read_csv(here("data", "prisoners2019.csv")) # Good
```

1. Use functions we've never covered.

```
total_prison_population <- aggregate(Population ~ State, df, sum)
```

Week 9: *Data Wrangling*

1. Selecting & filtering
2. Sequences with pipes

BREAK

3. Creating new variables
4. Grouped operations

Week 9: *Data Wrangling*

1. **Selecting & filtering**

2. Sequences with pipes

BREAK

3. Creating new variables

4. Grouped operations

Before we start

Make sure you have the "tidyverse" installed

```
install.packages('tidyverse')
```

(this is at the top of the `practice.R` file)

Remember: you only need to install packages once!

The tidyverse: `stringr` + `dplyr` + `readr` + `ggplot2` + ...



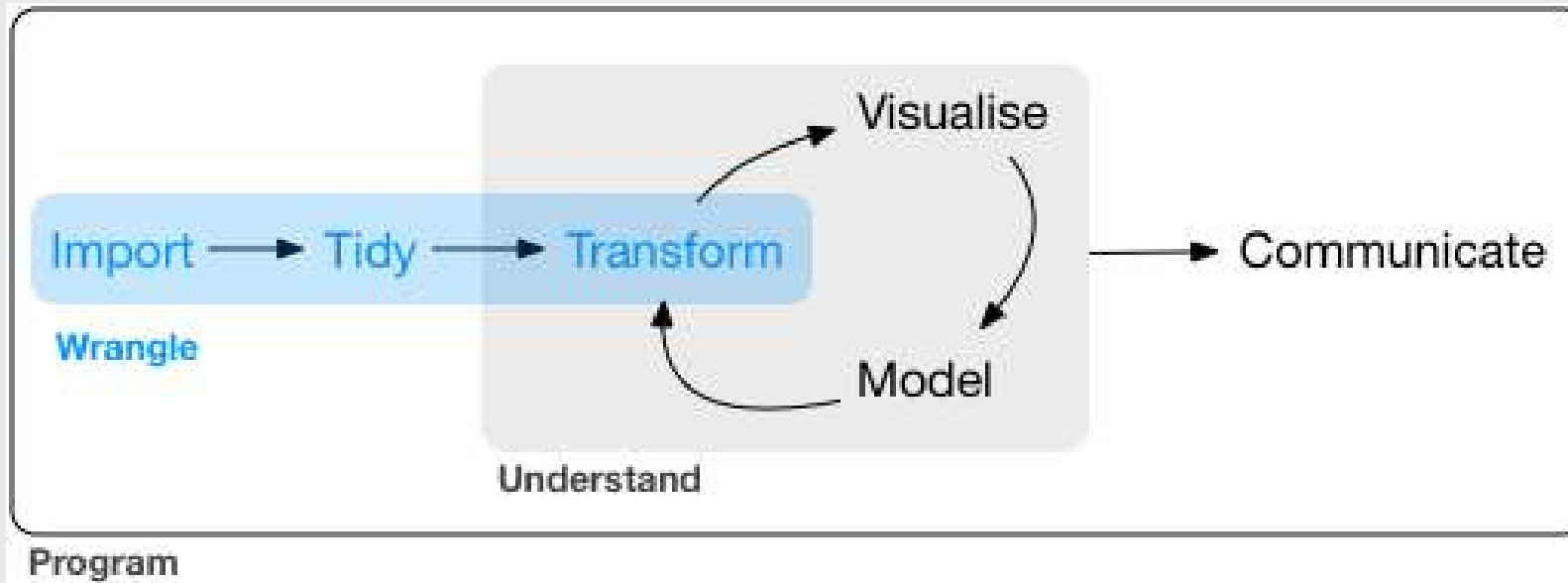
Art by [Allison Horst](#)

Today: better data wrangling with **dplyr**



Art by [Allison Horst](#)

80% of the job is data wrangling



The main `dplyr` "verbs"

- `select()`: subset columns
- `filter()`: subset rows on conditions
- `arrange()`: sort data frame
- `mutate()`: create new columns by using information from other columns
- `group_by()`: group data to perform grouped operations
- `summarize()`: create summary statistics (usually on grouped data)
- `count()`: count discrete rows

This week's British Band: **The Spice Girls**

```
spicegirls <- tibble(  
  firstName    = c("Melanie", "Melanie", "Emma", "Geri", "Victoria"),  
  lastName     = c("Brown", "Chisholm", "Bunton", "Halliwell", "Beckham"),  
  spice        = c("Scary", "Sporty", "Baby", "Ginger", "Posh"),  
  yearOfBirth  = c(1975, 1974, 1976, 1972, 1974),  
  deceased     = c(FALSE, FALSE, FALSE, FALSE, FALSE)  
)  
spicegirls
```

```
#> # A tibble: 5 × 5  
#>   firstName lastName  spice yearOfBirth deceased  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>  
#> 1 Melanie   Brown     Scary        1975 FALSE  
#> 2 Melanie   Chisholm Sporty        1974 FALSE  
#> 3 Emma      Bunton    Baby         1976 FALSE  
#> 4 Geri      Halliwell Ginger       1972 FALSE  
#> 5 Victoria Beckham  Posh         1974 FALSE
```

Select columns with `select()`

Subset Variables (Columns)



Select columns with `select()`

Example: Select the columns `firstName` & `lastName`

Base R:

```
spicegirls[c('firstName', 'lastName')]
```

```
#> # A tibble: 5 × 2
#>   firstName lastName
#>   <chr>      <chr>
#> 1 Melanie    Brown
#> 2 Melanie    Chisholm
#> 3 Emma       Bunton
#> 4 Geri       Halliwell
#> 5 Victoria   Beckham
```

Select columns with `select()`

Example: Select the columns `firstName` & `lastName`

dplyr: (note that you don't need `""` around names)

```
select(spicegirls, firstName, lastName)
```

```
#> # A tibble: 5 × 2
#>   firstName lastName
#>   <chr>      <chr>
#> 1 Melanie    Brown
#> 2 Melanie    Chisholm
#> 3 Emma       Bunton
#> 4 Geri       Halliwell
#> 5 Victoria   Beckham
```

Select columns with `select()`

Use the `-` sign to drop columns:

```
select(spicegirls, -firstName, -lastName)
```

```
#> # A tibble: 5 × 3  
#>   spice  yearOfBirth deceased  
#>   <chr>      <dbl> <lgl>  
#> 1 Scary      1975 FALSE  
#> 2 Sporty     1974 FALSE  
#> 3 Baby       1976 FALSE  
#> 4 Ginger     1972 FALSE  
#> 5 Posh       1974 FALSE
```

Select columns with `select()`

Select columns based on name criteria:

- `ends_with()` = Select columns that end with a character string
- `contains()` = Select columns that contain a character string
- `matches()` = Select columns that match a regular expression
- `one_of()` = Select column names that are from a group of names

Select columns with `select()`

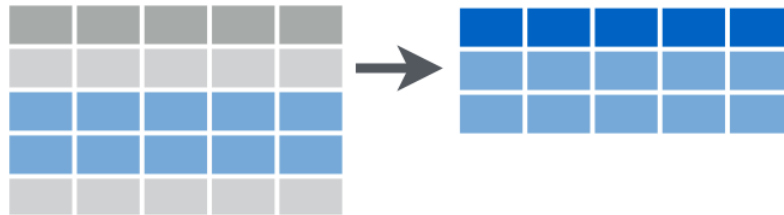
Select only the "name" columns

```
select(spicegirls, ends_with('name'))
```

```
#> # A tibble: 5 × 2  
#>   firstName lastName  
#>   <chr>      <chr>  
#> 1 Melanie    Brown  
#> 2 Melanie    Chisholm  
#> 3 Emma       Bunton  
#> 4 Geri       Halliwell  
#> 5 Victoria   Beckham
```

Select rows with `filter()`

Subset Observations (Rows)



Select rows with `filter()`

Example: Filter the band members born after 1974

```
## # A tibble: 5 x 5
##   firstName lastName  spice yearOfBirth deceased
##   <chr>      <chr>    <chr>      <dbl> <lgl>
## 1 Melanie   Brown     Scary      1975 FALSE
## 2 Melanie   Chisholm Sporty      1974 FALSE
## 3 Emma      Bunton    Baby       1976 FALSE
## 4 Geri       Halliwell Ginger     1972 FALSE
## 5 Victoria  Beckham   Posh       1974 FALSE
```

Select rows with `filter()`

Example: Filter the band members born after 1974

Base R:

```
spicegirls[spicegirls$yearOfBirth > 1974,]
```

```
#> # A tibble: 2 × 5  
#>   firstName lastName spice yearOfBirth deceased  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>  
#> 1 Melanie   Brown     Scary        1975 FALSE  
#> 2 Emma      Bunton    Baby         1976 FALSE
```

Select rows with `filter()`

Example: Filter the band members born after 1974

dplyr:

```
filter(spicegirls, yearOfBirth > 1974)
```

```
#> # A tibble: 2 × 5  
#>   firstName lastName spice yearOfBirth deceased  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>  
#> 1 Melanie   Brown    Scary        1975 FALSE  
#> 2 Emma      Bunton   Baby         1976 FALSE
```

Select rows with `filter()`

Example: Filter the band members born after 1974 **& are named "Melanie"**

dplyr:

```
filter(spicegirls, yearOfBirth > 1974 & firstName == "Melanie")
```

```
#> # A tibble: 1 × 5  
#>   firstName lastName spice yearOfBirth deceased  
#>   <chr>      <chr>   <chr>      <dbl> <lgl>  
#> 1 Melanie   Brown    Scary      1975  FALSE
```

Logic operators for `filter()`

Description	Example
Values greater than 1	<code>value > 1</code>
Values greater than or equal to 1	<code>value >= 1</code>
Values less than 1	<code>value < 1</code>
Values less than or equal to 1	<code>value <= 1</code>
Values equal to 1	<code>value == 1</code>
Values not equal to 1	<code>value != 1</code>
Values in the set <code>c(1, 4)</code>	<code>value %in% c(1, 4)</code>

Removing missing values

Drop all rows where `variable` is `NA`

```
filter(data, !is.na(variable))
```


Your turn: wildlife impacts data

10:00

1) Create the data frame object `df` by using `here()` and `read_csv()` to load the `wildlife_impacts.csv` file in the `data` folder.

2) Use the `df` object and the `select()` and `filter()` functions to answer the following questions:

- Create a new data frame, `df_birds`, that contains only the variables (columns) about the species of bird.
- Create a new data frame, `dc`, that contains only the observations (rows) from DC airports.
- Create a new data frame, `dc_birds_known`, that contains only the observations (rows) from DC airports and those where the species of bird is known.
- How many *known* unique species of birds have been involved in accidents at DC airports?

Week 9: *Data Wrangling*

1. Selecting & filtering
2. Sequences with pipes

BREAK

3. Creating new variables
4. Grouped operations

Create sequences of operations with "pipes"



[The Treachery of Images](#), René Magritte



[magrittr package](#)

Think of %>% as the words "...and then..."

Without Pipes (read from inside-out):

```
leave_house(get_dressed(get_out_of_bed(wake_up(me))))
```

With Pipes:

```
me %>%  
  wake_up() %>%  
  get_out_of_bed() %>%  
  get_dressed() %>%  
  leave_house()
```

Sequence operations with pipes: %>%

1. Filter the band members born after 1974
2. Select only the columns `firstName` & `lastName`

Without Pipes:

```
select(filter(spicegirls, yearOfBirth > 1974), firstName, lastName)
```

```
#> # A tibble: 2 × 2  
#>   firstName lastName  
#>   <chr>      <chr>  
#> 1 Melanie   Brown  
#> 2 Emma      Bunton
```

Sequence operations with pipes: %>%

1. Filter the band members born after 1974
2. Select only the columns `firstName` & `lastName`

With Pipes:

```
spicegirls %>%  
  filter(yearOfBirth > 1974) %>%  
  select(firstName, lastName)
```

```
#> # A tibble: 2 × 2  
#>   firstName lastName  
#>   <chr>      <chr>  
#> 1 Melanie   Brown  
#> 2 Emma      Bunton
```

Think of the words "...and then..."

Without Pipes:

```
select(filter(spicegirls, yearOfBirth > 1974), firstName, lastName)
```

With Pipes: Note that you don't need to repeat the dataframe name

```
spicegirls %>%  
  filter(yearOfBirth > 1974) %>%  
  select(firstName, lastName)
```

Sort rows with `arrange()`

Sort the data frame by year of birth:

```
spicegirls %>%  
  arrange(yearOfBirth)
```

```
#> # A tibble: 5 × 5  
#>   firstName lastName  spice yearOfBirth deceased  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>  
#> 1 Geri      Halliwell Ginger      1972 FALSE  
#> 2 Melanie   Chisholm  Sporty      1974 FALSE  
#> 3 Victoria Beckham  Posh        1974 FALSE  
#> 4 Melanie   Brown     Scary       1975 FALSE  
#> 5 Emma      Bunton    Baby        1976 FALSE
```


Sort rows with `arrange()`

Use the `desc()` function to sort in descending order:

```
spicegirls %>%  
  arrange(desc(yearOfBirth))
```

```
#> # A tibble: 5 × 5  
#>   firstName lastName  spice yearOfBirth deceased  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>  
#> 1 Emma      Bunton    Baby        1976 FALSE  
#> 2 Melanie   Brown     Scary        1975 FALSE  
#> 3 Melanie   Chisholm Sporty        1974 FALSE  
#> 4 Victoria Beckham  Posh         1974 FALSE  
#> 5 Geri       Halliwell Ginger        1972 FALSE
```

Sort rows with `arrange()`

Example of filtering, arranging, and selecting:

```
spicegirls %>%  
  filter(yearOfBirth < 1975) %>%  
  arrange(desc(yearOfBirth)) %>%  
  select(ends_with('name'))
```

```
#> # A tibble: 3 × 2  
#>   firstName lastName  
#>   <chr>      <chr>  
#> 1 Melanie   Chisholm  
#> 2 Victoria Beckham  
#> 3 Geri      Halliwell
```

Your turn

10:00

- 1) Create the data frame object `df` by using `here()` and `read_csv()` to load the `wildlife_impacts.csv` file in the `data` folder.
- 2) Use the `df` object and `select()`, `filter()`, and `%>%` to answer the following questions:
 - Create a new data frame, `dc_dawn`, that contains only the observations (rows) from DC airports that occurred at dawn.
 - Create a new data frame, `dc_dawn_birds`, that contains only the observations (rows) from DC airports that occurred at dawn and only the variables (columns) about the species of bird.
 - Create a new data frame, `dc_dawn_birds_known`, that contains only the observations (rows) from DC airports that occurred at dawn and only the variables (columns) about the KNOWN species of bird.
 - How many *known* unique species of birds have been involved in accidents at DC airports at dawn?

Break

05 : 00

Week 9: *Data Wrangling*

1. Selecting & filtering
2. Sequences with pipes

BREAK

3. Creating new variables
4. Grouped operations

Create new variables with `mutate()`

Make New Variables





Create new variables with `mutate()`

Example: Use the `yearOfBirth` variable to compute the age of each band member

Base R:

```
spicegirls$age <- 2022 - spicegirls$yearOfBirth
```

dplyr:

```
spicegirls %>%  
  mutate(age = 2022 - yearOfBirth)
```

```
#> # A tibble: 5 × 6  
#>   firstName lastName  spice yearOfBirth deceased  age  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>    <dbl>  
#> 1 Melanie   Brown      Scary        1975 FALSE     47  
#> 2 Melanie   Chisholm   Sporty       1974 FALSE     48  
#> 3 Emma      Bunton     Baby        1976 FALSE     46  
#> 4 Geri       Halliwell  Ginger       1972 FALSE     50  
#> 5 Victoria  Beckham    Posh         1971 FALSE     48
```


You can *immediately* use new variables

```
spicegirls %>%  
  mutate(  
    age = 2022 - yearOfBirth,  
    meanAge = mean(age)) # Immediately using the "age" variable
```

```
#> # A tibble: 5 × 7  
#>   firstName lastName  spice yearOfBirth deceased   age meanAge  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>    <dbl>  <dbl>  
#> 1 Melanie    Brown     Scary      1975 FALSE    47     47.8  
#> 2 Melanie    Chisholm Sporty     1974 FALSE    48     47.8  
#> 3 Emma       Bunton    Baby       1976 FALSE    46     47.8  
#> 4 Geri       Halliwell Ginger     1972 FALSE    50     47.8  
#> 5 Victoria   Beckham   Posh       1974 FALSE    48     47.8
```

Handling if/else conditions

`ifelse(<condition>, <if TRUE>, <else>)`

```
spicegirls %>%  
  mutate(  
    yobAfter74 = ifelse(yearOfBirth > 1974, "yes", "no"))
```

```
#> # A tibble: 5 × 6  
#>   firstName lastName  spice yearOfBirth deceased yobAfter74  
#>   <chr>      <chr>    <chr>      <dbl> <lgl>      <chr>  
#> 1 Melanie   Brown     Scary      1975 FALSE     yes  
#> 2 Melanie   Chisholm Sporty     1974 FALSE     no  
#> 3 Emma      Bunton    Baby      1976 FALSE     yes  
#> 4 Geri       Halliwell Ginger     1972 FALSE     no  
#> 5 Victoria  Beckham   Posh      1974 FALSE     no
```

Your turn

10:00

- 1) Create the data frame object `df` by using `here()` and `read_csv()` to load the `wildlife_impacts.csv` file in the `data` folder.
- 2) Use the `df` object with `%>%` and `mutate()` to create the following new variables:
 - `height_miles`: The `height` variable converted to miles (Hint: there are 5,280 feet in a mile).
 - `cost_mil`: Is `TRUE` if the repair costs was greater or equal to \$1 million, `FALSE` otherwise.
 - `season`: One of four seasons based on the `incident_month` variable:
 - `spring`: March, April, May
 - `summer`: June, July, August
 - `fall`: September, October, November
 - `winter`: December, January, February

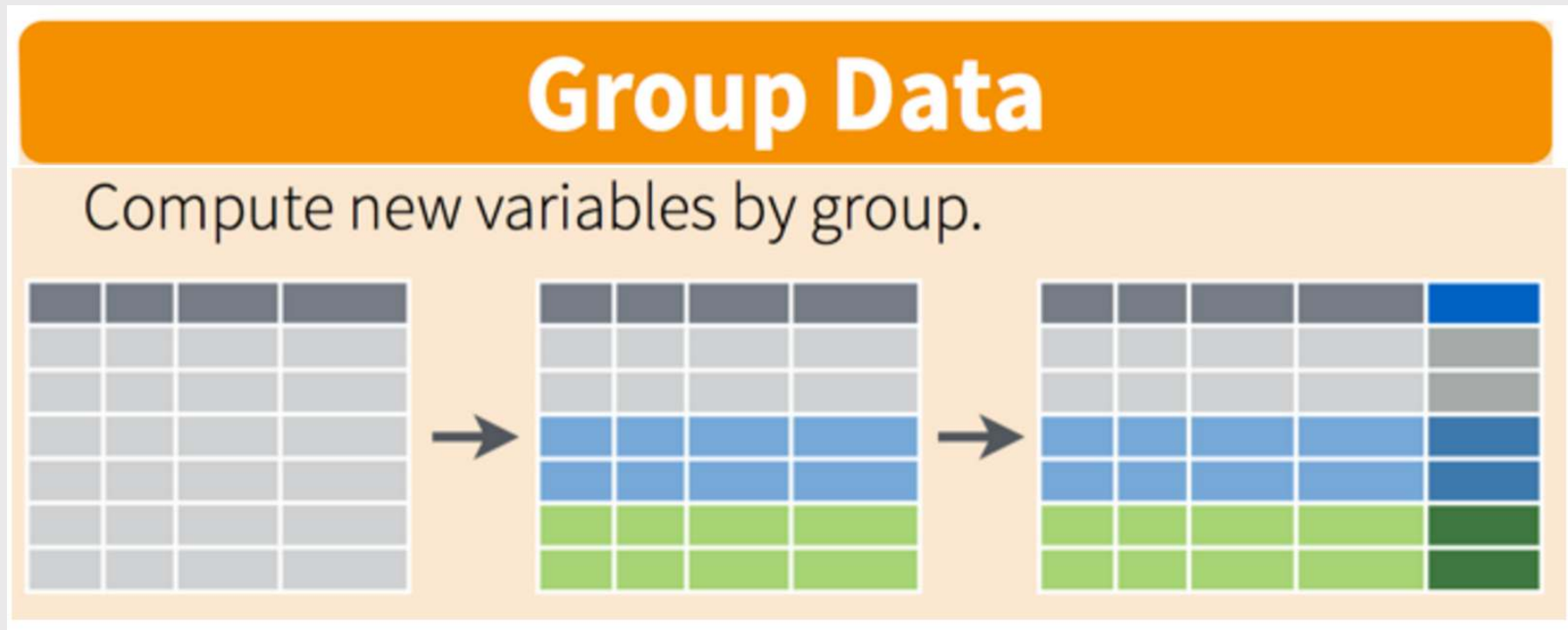
Week 9: *Data Wrangling*

1. Selecting & filtering
2. Sequences with pipes

BREAK

3. Creating new variables
4. Grouped operations

Compute by group with `group_by`



Compute by group with `group_by`

```
bands
```

```
#> # A tibble: 9 × 5  
#>   firstName lastName yearOfBirth deceased band  
#>   <chr>      <chr>      <dbl> <lgl>    <chr>  
#> 1 Melanie    Brown      1975 FALSE    spicegirls  
#> 2 Melanie    Chisholm   1974 FALSE    spicegirls  
#> 3 Emma       Bunton    1976 FALSE    spicegirls  
#> 4 Geri       Halliwell  1972 FALSE    spicegirls  
#> 5 Victoria   Beckham   1974 FALSE    spicegirls  
#> 6 John       Lennon    1940 TRUE     beatles  
#> 7 Paul       McCartney  1942 FALSE    beatles  
#> 8 Ringo      Starr     1940 FALSE    beatles  
#> 9 George     Harrison  1943 TRUE     beatles
```

Compute by group with `group_by`

Compute the mean band member age across the whole dataset

```
bands %>%  
  mutate(  
    age = 2024 - yearOfBirth,  
    mean_age = mean(age) # This is the mean across both bands  
  )
```

```
#> # A tibble: 9 × 7  
#>   firstName lastName yearOfBirth deceased band      age mean_age  
#>   <chr>      <chr>      <dbl> <lgl>    <chr>    <dbl>    <dbl>  
#> 1 Melanie   Brown      1975 FALSE    spicegirls 49      64.4  
#> 2 Melanie   Chisholm   1974 FALSE    spicegirls 50      64.4  
#> 3 Emma      Bunton     1976 FALSE    spicegirls 48      64.4  
#> 4 Geri       Halliwell  1972 FALSE    spicegirls 52      64.4  
#> 5 Victoria  Beckham    1974 FALSE    spicegirls 50      64.4  
#> 6 John       Lennon     1940 TRUE     beatles    84      64.4  
#> 7 Paul       McCartney  1942 FALSE    beatles    82      64.4  
#> 8 Ringo      Starr      1940 FALSE    beatles    84      64.4  
#> 9 George     Harrison   1943 TRUE     beatles    81      64.4
```

Compute by group with `group_by`

Compute the mean band member age **for each band**

```
bands %>%  
  mutate(age = 2024 - yearOfBirth) %>%  
  group_by(band) %>% # Everything after this will be done each band  
  mutate(mean_age = mean(age))
```

```
#> # A tibble: 9 × 7  
#> # Groups:   band [2]  
#>   firstName lastName yearOfBirth deceased band      age mean_age  
#>   <chr>      <chr>      <dbl> <lgl>    <chr>    <dbl>    <dbl>  
#> 1 Melanie   Brown      1975 FALSE    spicegirls 49      49.8  
#> 2 Melanie   Chisholm   1974 FALSE    spicegirls 50      49.8  
#> 3 Emma      Bunton     1976 FALSE    spicegirls 48      49.8  
#> 4 Geri       Halliwell  1972 FALSE    spicegirls 52      49.8  
#> 5 Victoria  Beckham    1974 FALSE    spicegirls 50      49.8  
#> 6 John      Lennon     1940 TRUE     beatles    84      82.8  
#> 7 Paul      McCartney  1942 FALSE    beatles    82      82.8  
#> 8 Ringo     Starr      1940 FALSE    beatles    84      82.8  
#> 9 George    Harrison   1943 TRUE     beatles    81      82.8
```


Summarize data frames with `summarise()`

Summarise Data



Summarize data frames with `summarise()`

Compute the mean band member age across the whole dataset

```
bands %>%  
  mutate(age = 2024 - yearOfBirth) %>%  
  summarise(mean_age = mean(age))
```

```
#> # A tibble: 1 × 1  
#>   mean_age  
#>   <dbl>  
#> 1     64.4
```

Summarize data frames with `summarise()`

Compute the mean band member age for **each band**

```
bands %>%  
  mutate(age = 2024 - yearOfBirth) %>%  
  group_by(band) %>%  
  summarise(mean_age = mean(age))
```

```
#> # A tibble: 2 × 2  
#>   band      mean_age  
#>   <chr>      <dbl>  
#> 1 beatles      82.8  
#> 2 spicegirls   49.8
```

Summarize data frames with `summarise()`

Compute the mean, min, and max band member age for **each band**

```
bands %>%  
  mutate(age = 2024 - yearOfBirth) %>%  
  group_by(band) %>%  
  summarise(  
    mean_age = mean(age),  
    min_age = min(age),  
    max_age = max(age)  
  )
```

```
#> # A tibble: 2 × 4  
#>   band      mean_age min_age max_age  
#>   <chr>      <dbl>   <dbl>   <dbl>  
#> 1 beatles      82.8     81     84  
#> 2 spicegirls   49.8     48     52
```

Computing counts of observations with `n()`

How many members are in each band?

```
bands %>%  
  group_by(band) %>%  
  summarise(numMembers = n())
```

```
#> # A tibble: 2 × 2  
#>   band      numMembers  
#>   <chr>      <int>  
#> 1 beatles      4  
#> 2 spicegirls   5
```

If you only want a quick count, use `count()`

These do the same thing:

```
bands %>%  
  group_by(band) %>%  
  summarise(numMembers = n())
```

```
#> # A tibble: 2 × 2  
#>   band      numMembers  
#>   <chr>      <int>  
#> 1 beatles      4  
#> 2 spicegirls   5
```

```
bands %>%  
  count(band)
```

```
#> # A tibble: 2 × 2  
#>   band      n  
#>   <chr>    <int>  
#> 1 beatles      4  
#> 2 spicegirls   5
```

If you only want a quick count, use `count()`

You can count by combinations of variables

```
bands %>%  
  mutate(nameStartsWithG = str_detect(firstName, '^G')) %>%  
  count(band, nameStartsWithG)
```

```
#> # A tibble: 4 × 3  
#>   band      nameStartsWithG     n  
#>   <chr>      <lgl>          <int>  
#> 1 beatles    FALSE             3  
#> 2 beatles    TRUE              1  
#> 3 spicegirls FALSE             4  
#> 4 spicegirls TRUE              1
```

Your turn

10:00

1) Create the data frame object `df` by using `here()` and `read_csv()` to load the `wildlife_impacts.csv` file in the `data` folder.

2) Use the `df` object and `group_by()`, `summarise()`, `count()`, and `%>%` to answer the following questions:

- Create a summary data frame that contains the mean `height` for each different time of day. Then use `write_csv()` to save it as the file "height_summary.csv" in your "data" folder.
- Create a summary data frame that contains the maximum `cost_repairs_infl_adj` for each year. Then use `write_csv()` to save it as the file "cost_summary.csv" in your "data" folder.
- Which *month* has had the greatest number of reported incidents?
- Which *year* has had the greatest number of reported incidents?

HW 9

Make sure you install the package `nycflights13`

```
install.packages('nycflights13')
```

This package includes **5 data frames**:

```
airlines  
airports  
flights  
planes  
weather
```