Week 11: Data Analysis 2 - Data Wrangling

EMSE 6574, Section 11

John Helveston
November 04, 2019

Announcements

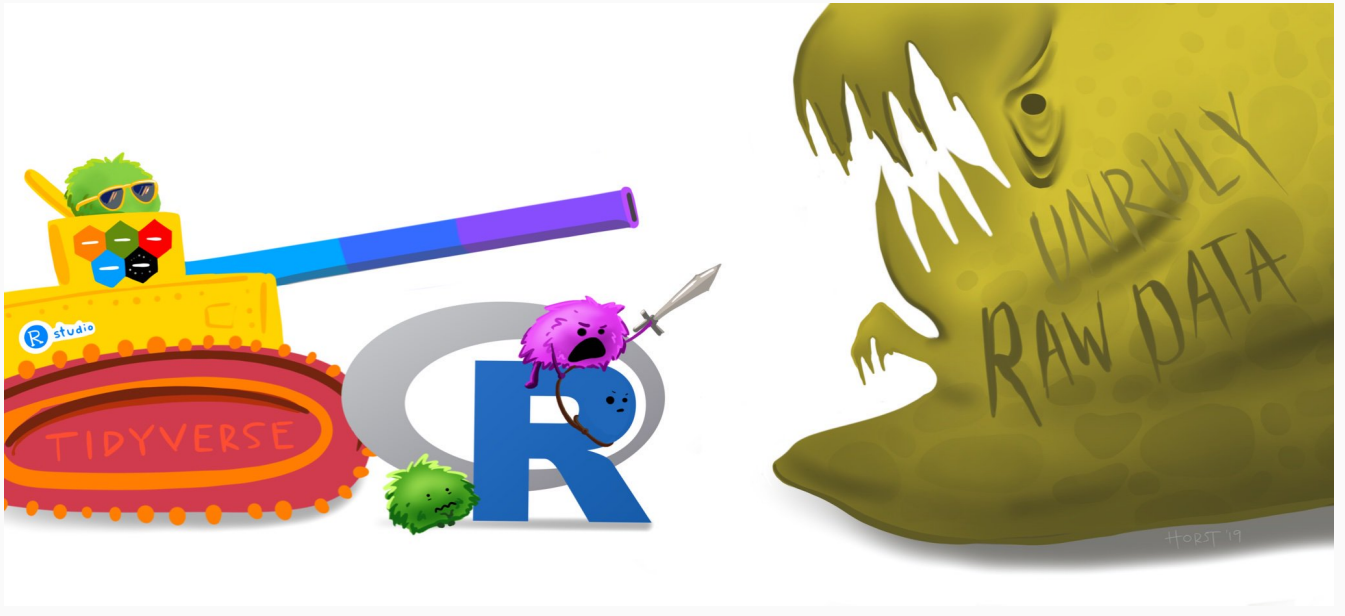
1) Download the `week11notes.zip` file for class today (link in `slack/classroom`).

2) Make sure you have the "tidyverse" installed:

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

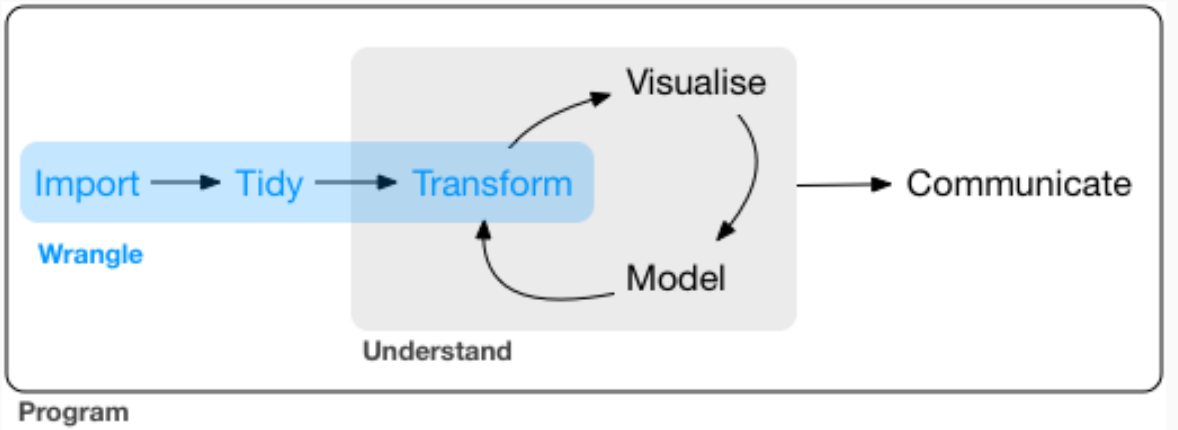
The tidyverse

`stringr` + `dplyr` + `readr` + `ggplot2` + more = `tidyverse`



Today: better data wrangling with **dplyr**

80% of the job is data wrangling



The main `dplyr` verbs

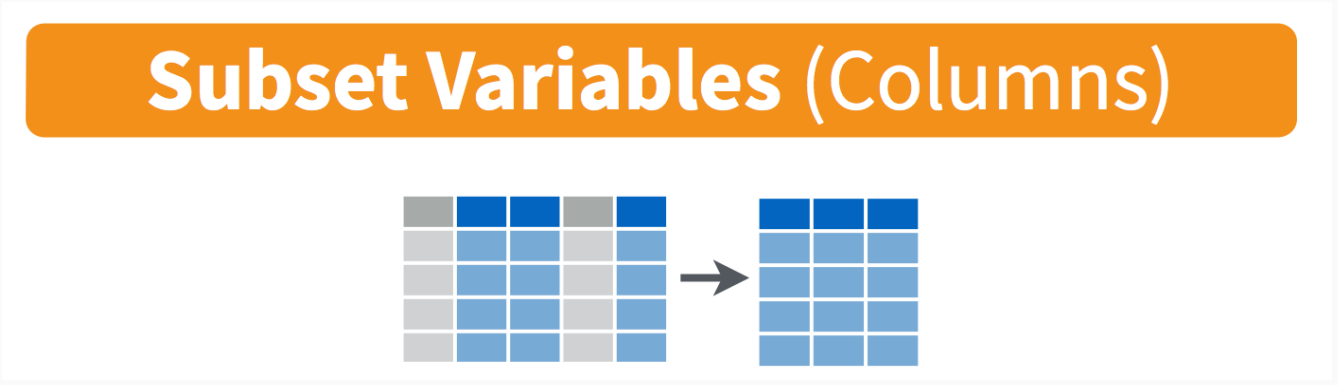
- `select()`: subset columns
- `filter()`: subset rows on conditions
- `arrange()`: sort results
- `mutate()`: create new columns by using information from other columns
- `group_by()`: group data to perform grouped operations
- `summarize()`: create summary statistics (usually on grouped data)
- `count()`: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



Select columns with `select()`

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

Select columns with `select()`

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

Base R:

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

```
## # A tibble: 5 x 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`

Base R:

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

dplyr:

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

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

Select columns with `select()`

Select all columns *except* certain ones with a `-` sign:

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

```
## # A tibble: 5 x 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:

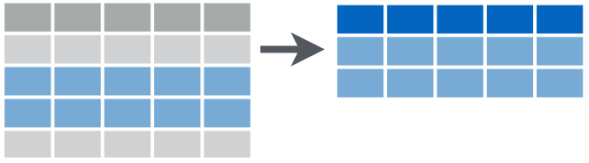
- `ends_with()` = Select columns that end with a character string
- `contains()` = Select columns that contain a character string
- `matches()` = Select columns that match a regular expression
- `one_of()` = Select column names that are from a group of names

```
# Select only the "name" columns
select(spicegirls, ends_with('name'))
```

```
## # A tibble: 5 x 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: 2 x 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

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

Base R:

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

Select rows with `filter()`

Example: Filter the band members born after 1974

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

Base R:

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

dplyr:

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

Select rows with `filter()`

Example: Filter the band members born after 1974

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

```
## # A tibble: 2 x 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

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

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

Example: Filter the band members named "Melanie"

```
filter(spicegirls, firstName == "Melanie")
```

```
## # A tibble: 2 x 5
##   firstName lastName spice  yearOfBirth deceased
##   <chr>     <chr>    <chr>        <dbl> <lgl>
## 1 Melanie   Brown    Scary         1975 FALSE
## 2 Melanie   Chisholm Sporty        1974 FALSE
```

Practice: `select` columns, `filter` rows

Data: Wildlife impacts data (we saw this last week)

1) Create the data frame object `df` by using `file.path()` and `read_csv()` to load the `wildlife_impacts.csv` file that is 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?

Sequence operations with pipes: `%>%`

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

Without Pipes:

```
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: `%>%`

What if I want to filter rows, and then select columns?

Sequence operations with pipes: `%>%`

What if I want to filter rows, and then select columns?

Example:

Step 1: Filter the band members born after 1974

Step 2: Select only the columns `firstName` & `lastName`

Without Pipes:

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

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

Sequence operations with pipes: `%>%`

What if I want to filter rows, and then select columns?

Example:

Step 1: Filter the band members born after 1974

Step 2: Select only the columns `firstName` & `lastName`

With Pipes:

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

```
## # A tibble: 2 x 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)
```

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

With Pipes:

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

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

Practice: `select`, `filter`, and `%>%`

Data: Wildlife impacts data

1) Create the data frame object `df` by using `file.path()` and `read_csv()` to load the `wildlife_impacts.csv` file that is 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, `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?

Sort rows with `arrange()`

Sort the data frame by year of birth:

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

```
## # A tibble: 5 x 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 x 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 x 2
##   firstName lastName
##   <chr>     <chr>
## 1 Melanie   Chisholm
## 2 Victoria  Beckham
## 3 Geri      Halliwell
```

5 minute break - stand up, move around,

5 minutes

Create new variables with `mutate()`

Create new variables with `mutate()`

Create new variables with `mutate()`

Example: Compute the age of each band member from `yearOfBirth`

Create new variables with `mutate()`

Example: Compute the age of each band member from `yearOfBirth`

Base R:

```
spicegirls$age <- 2019 - spicegirls$yearOfBirth
```

Create new variables with `mutate()`

Example: Compute the age of each band member from `yearOfBirth`

Base R:

```
spicegirls$age <- 2019 - spicegirls$yearOfBirth
```

dplyr:

```
spicegirls %>%
    mutate(age = 2019 - yearOfBirth)
```

```
## # A tibble: 5 x 6
##   firstName lastName  spice  yearOfBirth deceased   age
##   <chr>     <chr>     <chr>        <dbl> <lgl>    <dbl>
## 1 Melanie   Brown     Scary         1975 FALSE       44
## 2 Melanie   Chisholm  Sporty        1974 FALSE       45
## 3 Emma      Bunton    Baby          1976 FALSE       43
## 4 Geri      Halliwell Ginger        1972 FALSE       47
## 5 Victoria  Beckham   Posh          1974 FALSE       45
```

You can immediately use new variables

```
spicegirls %>%
    select(firstName, lastName, yearOfBirth) %>%
    mutate(
        age      = 2019 - yearOfBirth,
        meanAge  = mean(age),
        youngest = (age == min(age)),
        oldest   = (age == max(age)))
```

```
## # A tibble: 5 x 7
##   firstName lastName  yearOfBirth   age meanAge youngest oldest
##   <chr>     <chr>           <dbl> <dbl>   <dbl> <lgl>    <lgl>
## 1 Melanie   Brown            1975    44    44.8 FALSE    FALSE
## 2 Melanie   Chisholm         1974    45    44.8 FALSE    FALSE
## 3 Emma      Bunton           1976    43    44.8 TRUE     FALSE
## 4 Geri      Halliwell        1972    47    44.8 FALSE    TRUE
## 5 Victoria  Beckham          1974    45    44.8 FALSE    FALSE
```

if/else statements with `if_else()`

To create a new variable based on a condition, use `if_else()`

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

```
spicegirls %>%
    mutate(
        bornEvenOrOdd = if_else(yearOfBirth %% 2 == 0, 'even', 'odd'))
```

```
## # A tibble: 5 x 6
##   firstName lastName  spice  yearOfBirth deceased bornEvenOrOdd
##   <chr>     <chr>     <chr>        <dbl> <lgl>    <chr>
## 1 Melanie   Brown     Scary         1975 FALSE    odd
## 2 Melanie   Chisholm  Sporty        1974 FALSE    even
## 3 Emma      Bunton    Baby          1976 FALSE    even
## 4 Geri      Halliwell Ginger        1972 FALSE    even
## 5 Victoria  Beckham   Posh          1974 FALSE    even
```

Practice: `mutate`

Data: Wildlife impacts data

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

2) Use the `df` object and the `mutate()` functions to add the following new variables:

- `height_miles`: The `height` variable converted to miles (Hint: there are 5,280 feet in a mile).
- `cost_mil`: `TRUE` if the repair costs was greater or equal to $1 million, `FALSE` otherwise.

BONUS: Use the `incident_month` variable to create a new variable `season`, which takes one of four values based on the incident month:

- `spring`: March, April, May
- `summer`: June, July, August
- `fall`: September, October, November
- `winter`: December, January, February

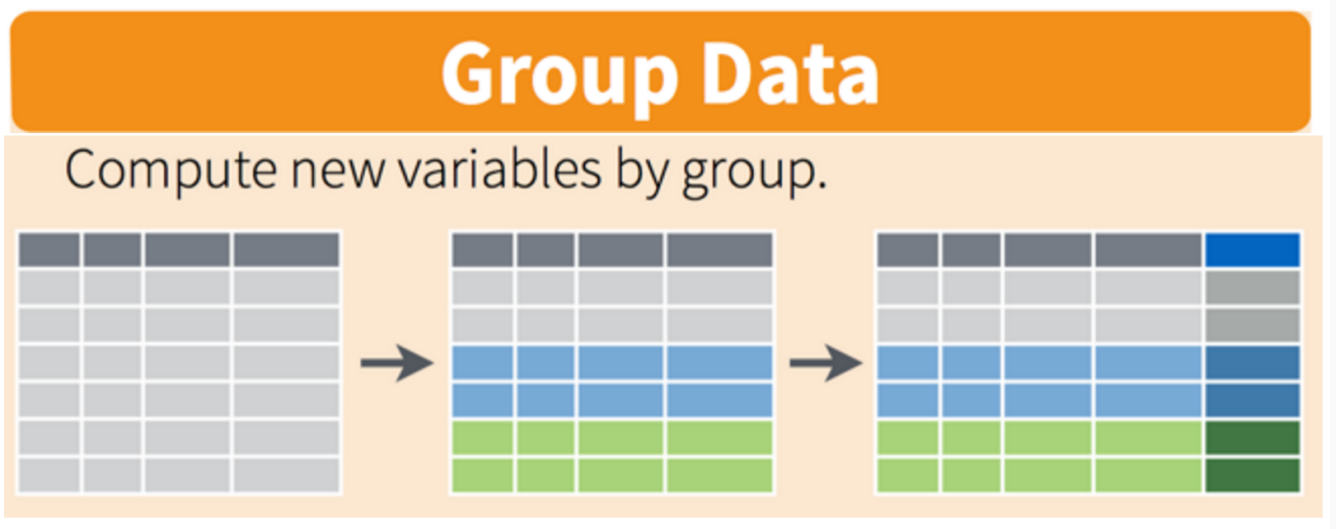
Split-apply-combine with `group_by`

1. **Split** the data into groups

2. **Apply** some analysis to each group

3. **Combine** the results

Split-apply-combine with `group_by`



Split-apply-combine with `group_by`

```
bands
```

```
## # A tibble: 9 x 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
```

Split-apply-combine with `group_by`

Compute the mean band member age for each band

```
bands %>%
    mutate(
        age = 2019 - yearOfBirth)
```

```
## # A tibble: 9 x 6
##   firstName lastName  yearOfBirth deceased band         age
##   <chr>     <chr>           <dbl> <lgl>    <chr>      <dbl>
## 1 Melanie   Brown            1975 FALSE    spicegirls    44
## 2 Melanie   Chisholm         1974 FALSE    spicegirls    45
## 3 Emma      Bunton           1976 FALSE    spicegirls    43
## 4 Geri      Halliwell        1972 FALSE    spicegirls    47
## 5 Victoria  Beckham          1974 FALSE    spicegirls    45
## 6 John      Lennon           1940 TRUE     beatles       79
## 7 Paul      McCartney        1942 FALSE    beatles       77
## 8 Ringo     Starr            1940 FALSE    beatles       79
## 9 George    Harrison         1943 TRUE     beatles       76
```

Split-apply-combine with `group_by`

Compute the mean band member age for each band

```
bands %>%
    mutate(
        age = 2019 - yearOfBirth,
        mean_age = mean(age))
```

```
## # A tibble: 9 x 7
##   firstName lastName  yearOfBirth deceased band         age mean_age
##   <chr>     <chr>           <dbl> <lgl>    <chr>      <dbl>    <dbl>
## 1 Melanie   Brown            1975 FALSE    spicegirls    44     59.4
## 2 Melanie   Chisholm         1974 FALSE    spicegirls    45     59.4
## 3 Emma      Bunton           1976 FALSE    spicegirls    43     59.4
## 4 Geri      Halliwell        1972 FALSE    spicegirls    47     59.4
## 5 Victoria  Beckham          1974 FALSE    spicegirls    45     59.4
## 6 John      Lennon           1940 TRUE     beatles       79     59.4
## 7 Paul      McCartney        1942 FALSE    beatles       77     59.4
## 8 Ringo     Starr            1940 FALSE    beatles       79     59.4
## 9 George    Harrison         1943 TRUE     beatles       76     59.4
```

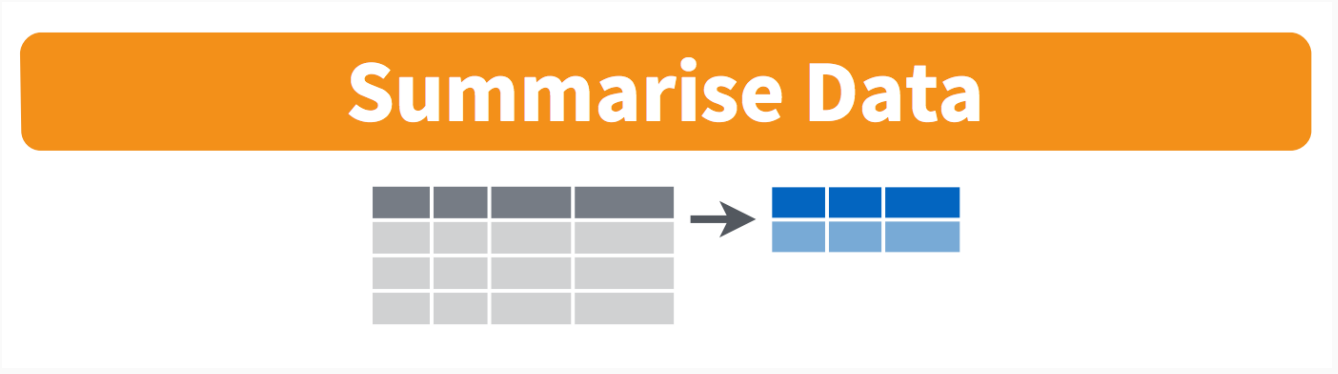
Split-apply-combine with `group_by`

Compute the mean band member age for each band

```
bands %>%
    mutate(age = 2019 - yearOfBirth) %>%
    group_by(band) %>%
    mutate(mean_age = mean(age))
```

```
## # A tibble: 9 x 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    44     44.8
## 2 Melanie   Chisholm         1974 FALSE    spicegirls    45     44.8
## 3 Emma      Bunton           1976 FALSE    spicegirls    43     44.8
## 4 Geri      Halliwell        1972 FALSE    spicegirls    47     44.8
## 5 Victoria  Beckham          1974 FALSE    spicegirls    45     44.8
## 6 John      Lennon           1940 TRUE     beatles       79     77.8
## 7 Paul      McCartney        1942 FALSE    beatles       77     77.8
## 8 Ringo     Starr            1940 FALSE    beatles       79     77.8
## 9 George    Harrison         1943 TRUE     beatles       76     77.8
```

Summarize data frames with `summarise()`



Summarize data frames with `summarise()`

Compute the mean band member age for each band

```
bands %>%
    mutate(age = 2019 - yearOfBirth) %>%
    group_by(band) %>%
    summarise(mean_age = mean(age))
```

```
## # A tibble: 2 x 2
##   band       mean_age
##   <chr>         <dbl>
## 1 beatles        77.8
## 2 spicegirls     44.8
```

Summarize data frames with `summarise()`

Compute the mean band member age for each band

```
bands %>%
    mutate(age = 2019 - yearOfBirth) %>%
    group_by(band) %>%
    summarise(
        mean_age = mean(age),
        min_age = min(age),
        max_age = max(age))
```

```
## # A tibble: 2 x 4
##   band       mean_age min_age max_age
##   <chr>         <dbl>   <dbl>   <dbl>
## 1 beatles        77.8      76      79
## 2 spicegirls     44.8      43      47
```

Practice: `group_by` + `summarise`

Data: Wildlife impacts data

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

2) Use the `df` object and the `group_by()` and `summarise` functions to answer the following questions:

- Create a summary data frame that contains the mean `height` for each different time of day.
- Create a summary data frame that contains the maximum `cost_repairs_infl_adj` for each year.

Count observations with `count()`

Example: How many members are in each band?

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

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

Count observations with `count()`

Same thing, but faster:

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

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

Count observations with `count()`

Counting *combinations of variables*:

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

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

Practice: `count`

Data: Wildlife impacts data

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

2) Use the `df` object and the `count()` function to answer the following questions:

- Which month has had the greatest number of reported incidents?
- Which year has had the greatest number of reported incidents?

Exporting data

Use `filePath()` + `write_csv()`

```
ageSummary <- bands %>%
    mutate(age = 2019 - yearOfBirth) %>%
    group_by(band) %>%
    summarise(
        mean_age = mean(age),
        min_age = min(age),
        max_age = max(age))
ageSummary
```

```
## # A tibble: 2 x 4
##   band       mean_age min_age max_age
##   <chr>         <dbl>   <dbl>   <dbl>
## 1 beatles        77.8      76      79
## 2 spicegirls     44.8      43      47
```

Exporting data

Save the `ageSummary` data frame in your "data" folder:

```
savePath <- file.path('data', 'ageSummary.csv')
write_csv(ageSummary, savePath)
```

HW 5

Make sure you install the package `nycflights13`

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

This package includes **5 data frames**:

```
airlines
airports
flights
planes
weather
```