

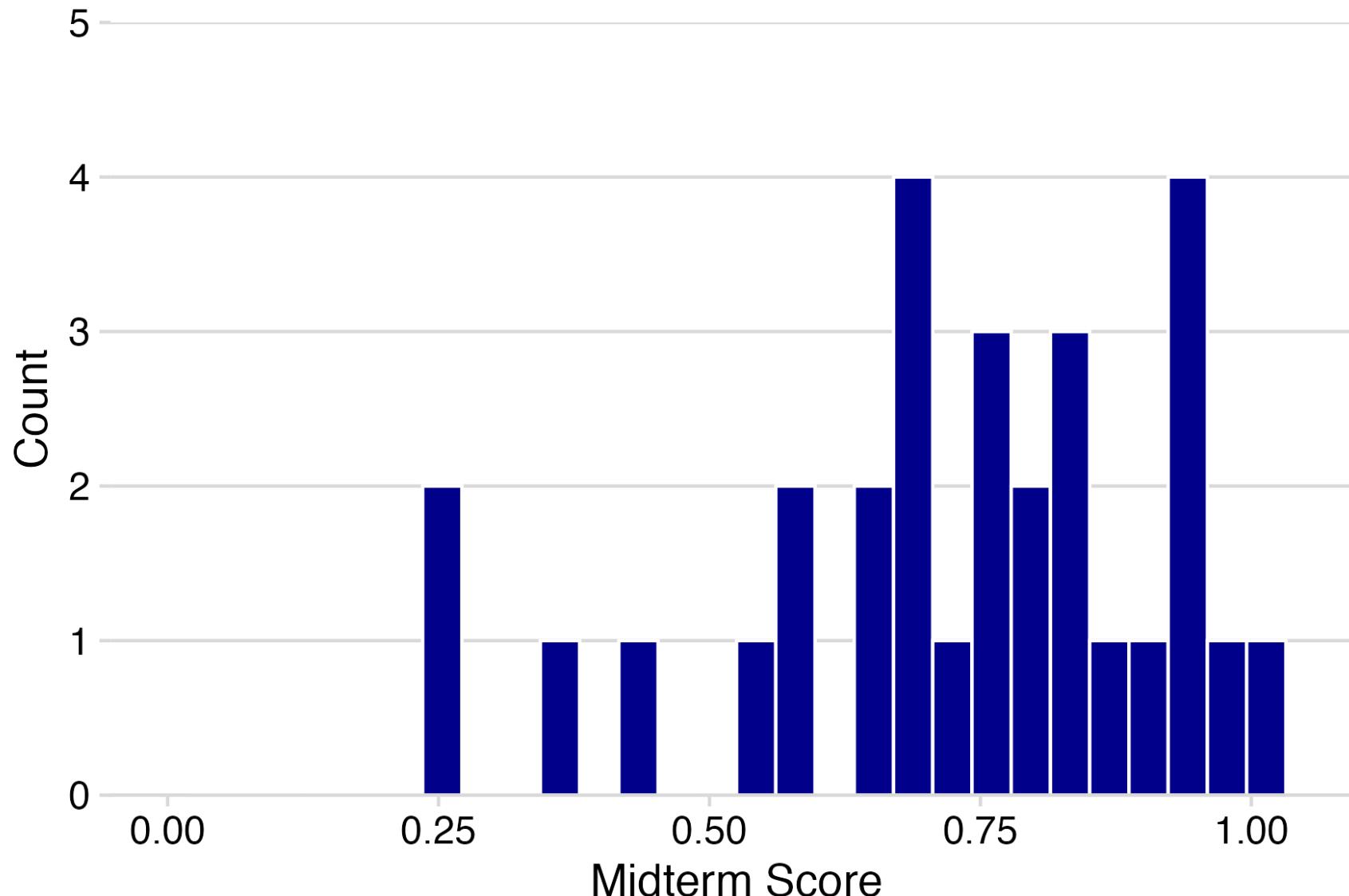


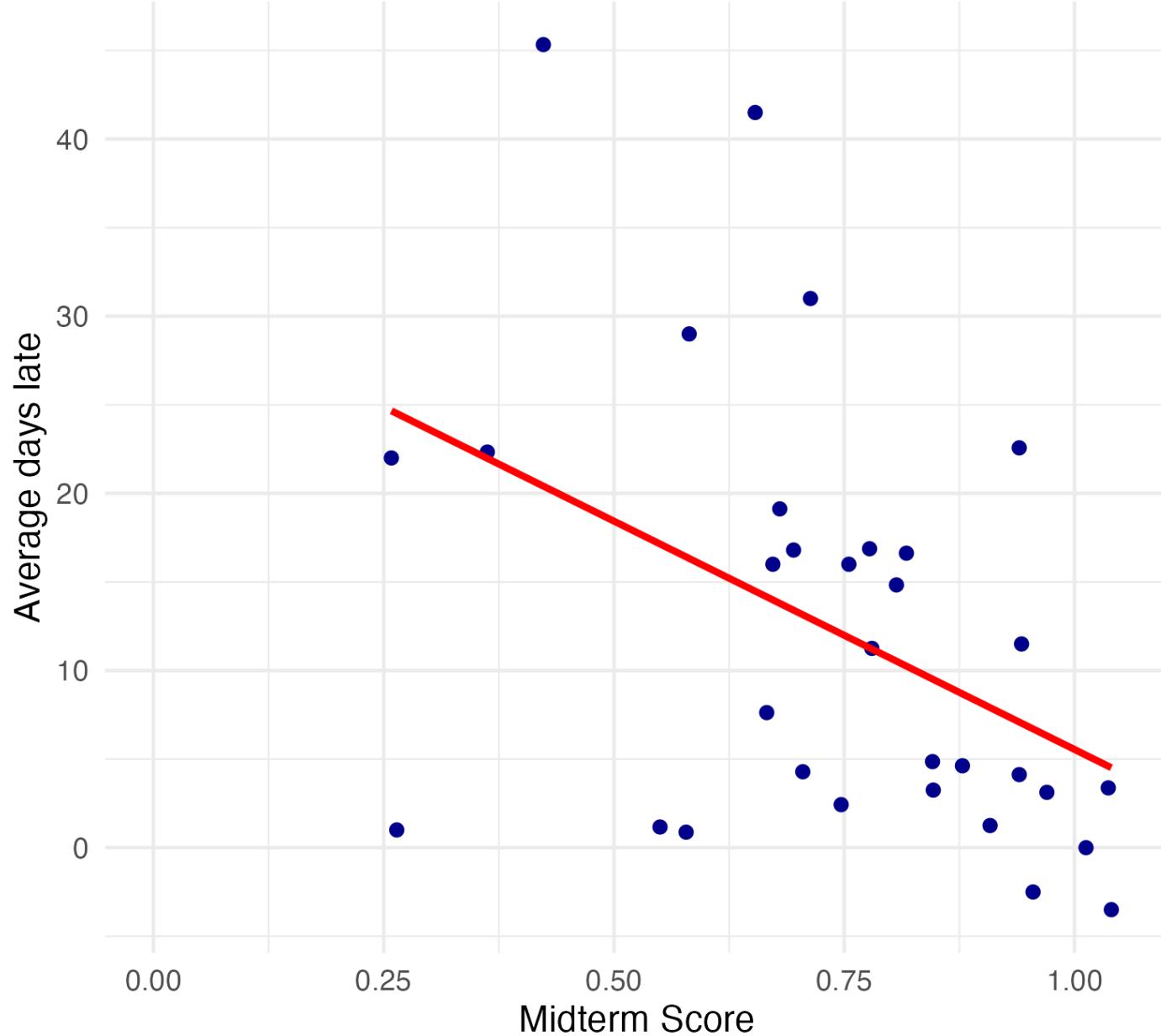
# Week 9: *Data Frames*

☰ EMSE 4571: Intro to Programming for Analytics

👤 John Paul Helveston

📅 March 24, 2022





# Revised late policy for HW 9-12

- Submissions by **6am** on due date: *full credit*
- Submissions by **6am** on following **Monday** (3 days late): *50% credit*
- Later submissions: *not graded* (i.e. a 0)

# Before we start

Make sure you have these packages installed and loaded:

```
install.packages("stringr")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("readr")
install.packages("here")
```

(At the top of the `notes_blank.R` file)

Remember: you only need to install them once!

"The purpose of computing  
is insight, not numbers"

- Richard Hamming



# Week 9: *Data Frames*

1. Basics

2. Slicing

BREAK

3. External data

# Week 9: *Data Frames*

1. Basics

2. Slicing

BREAK

3. External data

# The data frame...in Excel

A screenshot of Microsoft Excel showing a data frame for the Beatles. The data is displayed in a grid with columns A through H and rows 1 through 15. The columns are labeled with headers: A (firstName), B (lastName), C (instrument), D (yearOfBirth), and E (deceased). The data for the Beatles is as follows:

	A	B	C	D	E	F	G	H
1	firstName	lastName	instrument	yearOfBirth	deceased			
2	John	Lennon	guitar	1940	TRUE			
3	Paul	McCartney	bass	1942	FALSE			
4	Ringo	Starr	drums	1940	FALSE			
5	George	Harrison	guitar	1943	TRUE			
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								

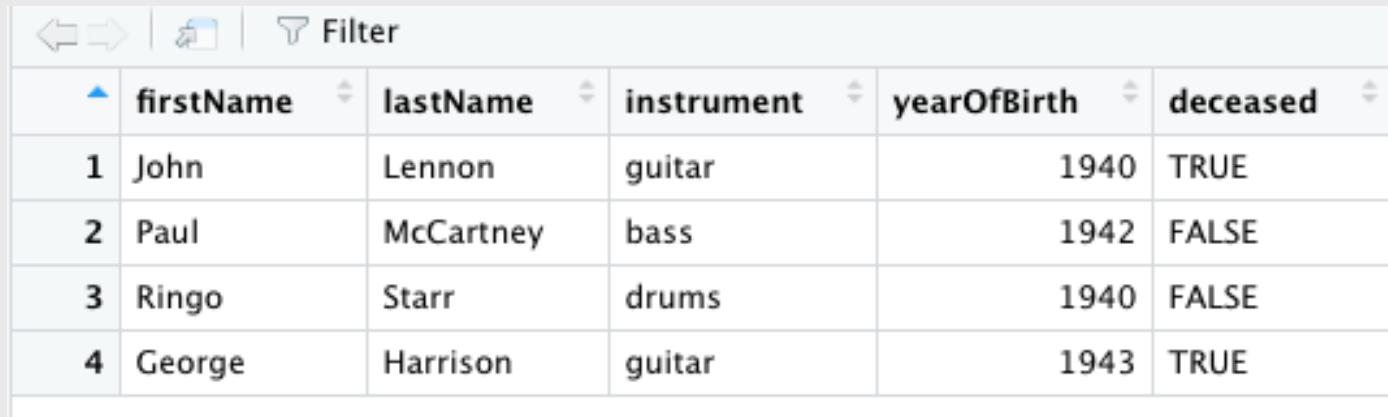
# The data frame...in R

```
beatles <- tibble(  
  firstName = c("John", "Paul", "Ringo", "George"),  
  lastName = c("Lennon", "McCartney", "Starr", "Harrison"),  
  instrument = c("guitar", "bass", "drums", "guitar"),  
  yearOfBirth = c(1940, 1942, 1940, 1943),  
  deceased = c(TRUE, FALSE, FALSE, TRUE)  
)  
  
beatles
```

```
#> # A tibble: 4 × 5  
#>   firstName lastName instrument yearOfBirth deceased  
#>   <chr>     <chr>      <chr>          <dbl> <lgl>  
#> 1 John       Lennon     guitar           1940  TRUE  
#> 2 Paul        McCartney bass            1942 FALSE  
#> 3 Ringo      Starr      drums           1940 FALSE  
#> 4 George     Harrison    guitar          1943  TRUE
```

# The data frame...in RStudio

```
view(beatles)
```



A screenshot of the RStudio View tool displaying a data frame named 'beatles'. The table has six columns: 'firstName', 'lastName', 'instrument', 'yearOfBirth', and 'deceased'. The 'firstName' column is sorted by row number (1, 2, 3, 4). The 'deceased' column contains logical values: TRUE for John Lennon and George Harrison, and FALSE for Paul McCartney and Ringo Starr.

	firstName	lastName	instrument	yearOfBirth	deceased
1	John	Lennon	guitar	1940	TRUE
2	Paul	McCartney	bass	1942	FALSE
3	Ringo	Starr	drums	1940	FALSE
4	George	Harrison	guitar	1943	TRUE

# Columns: Vectors of values (must be same data type)

```
beatles
```

```
#> # A tibble: 4 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>        <dbl>    <lgl>
#> 1 John       Lennon     guitar        1940    TRUE
#> 2 Paul       McCartney bass         1942    FALSE
#> 3 Ringo      Starr      drums        1940    FALSE
#> 4 George     Harrison   guitar        1943    TRUE
```

Extract a column using `$`

```
beatles$firstName
```

```
#> [1] "John"    "Paul"    "Ringo"   "George"
```

# Rows: Information about individual observations

Information about *John Lennon* is in the first row:

```
beatles[1,]
```

```
#> # A tibble: 1 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>        <dbl> <lgl>
#> 1 John       Lennon     guitar        1940  TRUE
```

Information about *Paul McCartney* is in the second row:

```
beatles[2,]
```

```
#> # A tibble: 1 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>        <dbl> <lgl>
#> 1 Paul       McCartney bass        1942 FALSE
```

# Make a data frame with `data.frame()`

```
beatles <- data.frame(  
  firstName = c("John", "Paul", "Ringo", "George"),  
  lastName = c("Lennon", "McCartney", "Starr", "Harrison"),  
  instrument = c("guitar", "bass", "drums", "guitar"),  
  yearOfBirth = c(1940, 1942, 1940, 1943),  
  deceased = c(TRUE, FALSE, FALSE, TRUE)  
)
```

```
beatles
```

```
#>   firstName lastName instrument yearOfBirth deceased  
#> 1      John    Lennon       guitar      1940     TRUE  
#> 2      Paul  McCartney      bass      1942    FALSE  
#> 3     Ringo      Starr      drums      1940    FALSE  
#> 4     George  Harrison       guitar      1943     TRUE
```

# Make a data frame with `tibble()`

```
library(dplyr)
```

```
beatles <- tibble(  
  firstName = c("John", "Paul", "Ringo", "George"),  
  lastName = c("Lennon", "McCartney", "Starr", "Harrison"),  
  instrument = c("guitar", "bass", "drums", "guitar"),  
  yearOfBirth = c(1940, 1942, 1940, 1943),  
  deceased = c(TRUE, FALSE, FALSE, TRUE)  
)
```

```
beatles
```

```
#> # A tibble: 4 × 5  
#>   firstName lastName instrument yearOfBirth deceased  
#>   <chr>     <chr>      <chr>          <dbl> <lgl>  
#> 1 John       Lennon      guitar        1940  TRUE  
#> 2 Paul       McCartney  bass         1942  FALSE  
#> 3 Ringo      Starr       drums        1940  FALSE  
#> 4 George     Harrison    guitar        1943  TRUE
```

# Why I use `tibble()` instead of `data.frame()`

1. The `tibble()` shows the **dimensions** and **data type**.
2. A tibble will only print the first few rows of data when you enter the object name  
Example: `faithful` vs. `as_tibble(faithful)`
3. Columns of class `character` are *never* converted into factors (don't worry about this for now...just know that tibbles make life easier when dealing with character type columns).

**Note:** I use the word "**data frame**" to refer to both `tibble()` and `data.frame()` objects

# Data frame vectors must have the same length

```
beatles <- tibble(  
  firstName = c("John", "Paul", "Ringo", "George", "Bob"), # Added "Bob"  
  lastName = c("Lennon", "McCartney", "Starr", "Harrison"),  
  instrument = c("guitar", "bass", "drums", "guitar"),  
  yearOfBirth = c(1940, 1942, 1940, 1943),  
  deceased = c(TRUE, FALSE, FALSE, TRUE)  
)
```

```
#> Error:  
#> ! Tibble columns must have compatible sizes.  
#> • Size 5: Existing data.  
#> • Size 4: Column `lastName`.  
#> i Only values of size one are recycled.
```

# Use NA for missing values

```
beatles <- tibble(  
  firstName = c("John", "Paul", "Ringo", "George", "Bob"),  
  lastName = c("Lennon", "McCartney", "Starr", "Harrison", NA), # Added NAs  
  instrument = c("guitar", "bass", "drums", "guitar", NA),  
  yearOfBirth = c(1940, 1942, 1940, 1943, NA),  
  deceased = c(TRUE, FALSE, FALSE, TRUE, NA)  
)
```

```
beatles
```

```
#> # A tibble: 5 × 5  
#>   firstName lastName instrument yearOfBirth deceased  
#>   <chr>     <chr>     <chr>          <dbl> <lgl>  
#> 1 John       Lennon    guitar        1940  TRUE  
#> 2 Paul       McCartney bass        1942 FALSE  
#> 3 Ringo      Starr     drums        1940 FALSE  
#> 4 George     Harrison  guitar        1943  TRUE  
#> 5 Bob        <NA>      <NA>           NA  NA
```

# Dimensions: `nrow()`, `ncol()`, & `dim()`

```
nrow(beatles) # Number of rows
```

```
#> [1] 5
```

```
ncol(beatles) # Number of columns
```

```
#> [1] 5
```

```
dim(beatles) # Number of rows and columns
```

```
#> [1] 5 5
```

# Use `names()` or `colnames()` to see the available variables

Get the names of columns:

```
names(beatles)
```

```
#> [1] "firstName"    "lastName"     "instrument"   "yearOfBirth"  "deceased"
```

```
colnames(beatles)
```

```
#> [1] "firstName"    "lastName"     "instrument"   "yearOfBirth"  "deceased"
```

Get the names of rows (rarely needed):

```
rownames(beatles)
```

```
#> [1] "1"  "2"  "3"  "4"  "5"
```

# Changing the column names

Change the column names with `names()` or `colnames()`:

```
names(beatles) <- c('one', 'two', 'three', 'four', 'five')
beatles
```

```
#> # A tibble: 5 × 5
#>   one     two     three     four     five
#>   <chr>   <chr>   <chr>   <dbl>   <lgl>
#> 1 John    Lennon   guitar  1940   TRUE
#> 2 Paul    McCartney bass   1942  FALSE
#> 3 Ringo   Starr    drums  1940  FALSE
#> 4 George  Harrison guitar  1943   TRUE
#> 5 Bob     <NA>    <NA>     NA   NA
```

# Changing the column names

Make all the column names upper-case:

```
colnames(beatles) <- stringr::str_to_upper(colnames(beatles))  
beatles
```

```
#> # A tibble: 5 × 5  
#>   FIRSTNAME LASTNAME  INSTRUMENT YEAROFBIRTH DECEASED  
#>   <chr>      <chr>      <chr>          <dbl> <lgl>  
#> 1 John       Lennon     guitar        1940  TRUE  
#> 2 Paul       McCartney bass           1942 FALSE  
#> 3 Ringo      Starr      drums        1940  FALSE  
#> 4 George     Harrison    guitar        1943  TRUE  
#> 5 Bob        <NA>       <NA>           NA  NA
```

# Combine data frames by columns using `bind_cols()`

Note: `bind_cols()` is from the **dplyr** library

```
names <- tibble(  
  firstName = c("John", "Paul", "Ringo", "George"),  
  lastName  = c("Lennon", "McCartney", "Starr", "Harrison"))  
  
instruments <- tibble(  
  instrument = c("guitar", "bass", "drums", "guitar"))
```

```
bind_cols(names, instruments)
```

```
#> # A tibble: 4 × 3  
#>   firstName lastName instrument  
#>   <chr>     <chr>    <chr>  
#> 1 John      Lennon    guitar  
#> 2 Paul      McCartney bass  
#> 3 Ringo    Starr     drums  
#> 4 George    Harrison  guitar
```

# Combine data frames by rows using `bind_rows()`

Note: `bind_rows()` is from the **dplyr** library

```
members1 <- tibble(  
  firstName = c("John", "Paul"),  
  lastName = c("Lennon", "McCartney"))  
  
members2 <- tibble(  
  firstName = c("Ringo", "George"),  
  lastName = c("Starr", "Harrison"))
```

```
bind_rows(members1, members2)
```

```
#> # A tibble: 4 × 2  
#>   firstName lastName  
#>   <chr>     <chr>  
#> 1 John      Lennon  
#> 2 Paul      McCartney  
#> 3 Ringo    Starr  
#> 4 George    Harrison
```

# Note: `bind_rows()` requires the **same** columns names:

```
colnames(members2) <- c("firstName", "LastName")
bind_rows(members1, members2)
```

```
#> # A tibble: 4 × 3
#>   firstName lastName LastName
#>   <chr>     <chr>     <chr>
#> 1 John       Lennon    <NA>
#> 2 Paul       McCartney <NA>
#> 3 Ringo      <NA>      Starr
#> 4 George     <NA>      Harrison
```

Note how <NA>s were created

08 : 00

## Your turn

Answer these questions using the `animals_farm` and `animals_pet` data frames:

1. Write code to find how many *rows* are in the `animals_farm` data frame?
2. Write code to find how many *columns* are in the `animals_pet` data frame?
3. Create a new data frame, `animals`, by combining `animals_farm` and `animals_pet`.
4. Change the column names of `animals` to title case.
5. Add a new column to `animals` called `type` that tells if an animal is a "farm" or "pet" animal.

# Week 9: *Data Frames*

1. Basics

2. **Slicing**

BREAK

3. External data

# Access data frame columns using the \$ symbol

```
beatles$firstName
```

```
#> [1] "John"    "Paul"    "Ringo"   "George"
```

```
beatles$lastName
```

```
#> [1] "Lennon"   "McCartney" "Starr"    "Harrison"
```

# Creating new variables with the \$ symbol

Add the hometown of the bandmembers:

```
beatles$hometown <- 'Liverpool'  
beatles
```

```
#> # A tibble: 4 × 6  
#>   firstName lastName instrument yearOfBirth deceased hometown  
#>   <chr>     <chr>    <chr>          <dbl> <lgl>    <chr>  
#> 1 John       Lennon    guitar        1940  TRUE    Liverpool  
#> 2 Paul       McCartney bass         1942 FALSE    Liverpool  
#> 3 Ringo      Starr     drums        1940 FALSE    Liverpool  
#> 4 George     Harrison  guitar        1943 TRUE    Liverpool
```

# Creating new variables with the \$ symbol

Add a new `alive` variable:

```
beatles$alive <- c(FALSE, TRUE, TRUE, FALSE)  
beatles
```

```
#> # A tibble: 4 × 7  
#>   firstName lastName instrument yearOfBirth deceased hometown alive  
#>   <chr>     <chr>    <chr>          <dbl> <lgl>    <chr>    <lgl>  
#> 1 John      Lennon    guitar        1940  TRUE    Liverpool FALSE  
#> 2 Paul      McCartney bass         1942  FALSE   Liverpool TRUE  
#> 3 Ringo     Starr     drums        1940  FALSE   Liverpool TRUE  
#> 4 George    Harrison  guitar       1943  TRUE    Liverpool FALSE
```

# You can compute new variables from current ones

Compute and add the age of the bandmembers:

```
beatles$age <- 2020 - beatles$yearOfBirth  
beatles
```

```
#> # A tibble: 4 × 8  
#>   firstName lastName instrument yearOfBirth deceased hometown alive   age  
#>   <chr>     <chr>    <chr>          <dbl> <lgl>    <chr>    <lgl> <dbl>  
#> 1 John       Lennon    guitar        1940  TRUE    Liverpool FALSE    80  
#> 2 Paul       McCartney bass         1942  FALSE   Liverpool TRUE     78  
#> 3 Ringo      Starr     drums        1940  FALSE   Liverpool TRUE     80  
#> 4 George     Harrison  guitar       1943  TRUE    Liverpool FALSE    77
```

# Access elements by index: DF [row, column]

General form for indexing elements:

```
DF [row, column]
```

Select the element in row 1, column 2:

```
beatles[1, 2]
```

```
#> # A tibble: 1 × 1  
#>   lastName  
#>   <chr>  
#> 1 Lennon
```

Select the elements in rows 1 & 2 and columns 2 & 3:

```
beatles[c(1, 2), c(2, 3)]
```

```
#> # A tibble: 2 × 2  
#>   lastName instrument  
#>   <chr>     <chr>  
#> 1 Lennon    guitar  
#> 2 McCartney bass
```

# Leave row or column "blank" to select all

```
beatles[c(1, 2),] # Selects all COLUMNS for rows 1 & 2
```

```
#> # A tibble: 2 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>        <dbl> <lgl>
#> 1 John       Lennon     guitar        1940  TRUE
#> 2 Paul       McCartney bass         1942 FALSE
```

```
beatles[,c(1, 2)] # Selects all ROWS for columns 1 & 2
```

```
#> # A tibble: 4 × 2
#>   firstName lastName
#>   <chr>     <chr>
#> 1 John       Lennon
#> 2 Paul       McCartney
#> 3 Ringo     Starr
#> 4 George    Harrison
```

# Negative indices exclude row / column

```
beatles[-1, ] # Select all ROWS except the first
```

```
#> # A tibble: 3 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>        <dbl> <lgl>
#> 1 Paul       McCartney bass            1942 FALSE
#> 2 Ringo      Starr      drums          1940 FALSE
#> 3 George     Harrison  guitar         1943 TRUE
```

```
beatles[,-1] # Select all COLUMNS except the first
```

```
#> # A tibble: 4 × 4
#>   lastName instrument yearOfBirth deceased
#>   <chr>      <chr>        <dbl> <lgl>
#> 1 Lennon     guitar        1940 TRUE
#> 2 McCartney  bass         1942 FALSE
#> 3 Starr      drums        1940 FALSE
#> 4 Harrison   guitar        1943 TRUE
```

# You can select columns by their names

Note: you don't need the comma to select an entire column

One column

```
beatles['firstName']
```

```
#> # A tibble: 4 × 1  
#>   firstName  
#>   <chr>  
#> 1 John  
#> 2 Paul  
#> 3 Ringo  
#> 4 George
```

Multiple columns

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

```
#> # A tibble: 4 × 2  
#>   firstName lastName  
#>   <chr>     <chr>  
#> 1 John      Lennon  
#> 2 Paul      McCartney  
#> 3 Ringo    Starr  
#> 4 George   Harrison
```

# Use logical indices to *filter* rows

## Which Beatles members are still alive?

Create a logical vector using the `deceased` column:

```
beatles$deceased == FALSE
```

```
#> [1] FALSE TRUE TRUE FALSE
```

Insert this logical vector in the ROW position of `beatles[, ]`:

```
beatles[beatles$deceased == FALSE, ]
```

```
#> # A tibble: 2 × 5
#>   firstName lastName instrument yearOfBirth deceased
#>   <chr>     <chr>      <chr>          <dbl> <lgl>
#> 1 Paul       McCartney bass            1942 FALSE
#> 2 Ringo      Starr      drums           1940 FALSE
```

# Your turn

10:00

Answer these questions using the `beatles` data frame:

1. Create a new column, `playsGuitar`, which is `TRUE` if the band member plays the guitar and `FALSE` otherwise.
2. Filter the data frame to select only the rows for the band members who have four-letter first names.
3. Create a new column, `fullName`, which contains the band member's first and last name separated by a space (e.g. "`John Lennon`")

*Break*

05 : 00

# Week 9: *Data Frames*

1. Basics

2. Slicing

BREAK

3. External data

# Getting data into R

Options:

1. Load external packages
2. Read in external files (usually a `.csv`\* file)

\*csv = "comma-separated values"

# Data from an R package

```
library(ggplot2)
```

See which data frames are available in a package:

```
data(package = "ggplot2")
```

# Find out about package data sets with ?

```
?msleep
```

```
msleep {ggplot2}
```

An updated and expanded version of the mammals sleep dataset

Description

This is an updated and expanded version of the mammals sleep dataset. Updated sleep times

# Previewing data frames: `msleep`

Look at the data in a "spreadsheet"-like way:

```
view(msleep)
```

This is "read-only" so you can't corrupt the data 😊

# My favorite quick summary: `glimpse()`

Preview each variable with `str()` or `glimpse()`

```
glimpse(msleep)
```

```
#> Rows: 83
#> Columns: 11
#> $ name      <chr> "Cheetah", "Owl monkey", "Mountain beaver", "Greater short-t
#> $ genus     <chr> "Acinonyx", "Aotus", "Aplopontia", "Blarina", "Bos", "Bradyp
#> $ vore       <chr> "carni", "omni", "herbi", "omni", "herbi", "herbi", "carni",
#> $ order      <chr> "Carnivora", "Primates", "Rodentia", "Soricomorpha", "Artiod
#> $ conservation <chr> "lc", NA, "nt", "lc", "domesticated", NA, "vu", NA, "domesti
#> $ sleep_total <dbl> 12.1, 17.0, 14.4, 14.9, 4.0, 14.4, 8.7, 7.0, 10.1, 3.0, 5.3,
#> $ sleep_rem   <dbl> NA, 1.8, 2.4, 2.3, 0.7, 2.2, 1.4, NA, 2.9, NA, 0.6, 0.8, 0.7
#> $ sleep_cycle <dbl> NA, NA, NA, 0.1333333, 0.6666667, 0.7666667, 0.3833333, NA,
#> $ awake        <dbl> 11.90, 7.00, 9.60, 9.10, 20.00, 9.60, 15.30, 17.00, 13.90, 2
#> $ brainwt     <dbl> NA, 0.01550, NA, 0.00029, 0.42300, NA, NA, NA, 0.07000, 0.09
#> $ bodywt      <dbl> 50.000, 0.480, 1.350, 0.019, 600.000, 3.850, 20.490, 0.045,
```

# Also very useful for quick checks: `head()` and `tail()`

View the **first** 6 rows with `head()`

```
head(msleep)
```

```
#> # A tibble: 6 × 11
#>   name                      genus
#>   <chr>                     <chr>
#> 1 Cheetah                   Acinonyx
#> 2 Owl monkey                Aotus
#> 3 Mountain beaver           Aplodonti
#> 4 Greater short-tailed shrew Blarina
#> 5 Cow                        Bos
#> 6 Three-toed sloth          Bradypus
```

View the **last** 6 rows with `tail()`

```
tail(msleep)
```

```
#> # A tibble: 6 × 11
#>   name                      genus    vore
#>   <chr>                     <chr>   <chr>
#> 1 Tenrec                    Tenrec   omni
#> 2 Tree shrew                Tupaia   omni
#> 3 Bottle-nosed dolphin     Tursiops carni
#> 4 Genet                      Genetta carni
#> 5 Arctic fox                 Vulpes   carni
#> 6 Red fox                   Vulpes   carni
```

# Importing an external data file

Note the `data.csv` file in your `data` folder.

- **DO NOT** double-click it!
- **DO NOT** open it in Excel!

Excel can **corrupt** your data!

If you **must** open it in Excel:

- Make a copy
- Open the copy

# Steps to importing external data files

## 1. Create a path to the data

```
library(here)
pathToData <- here('data', 'data.csv')
pathToData
```

```
#> [1] "/Users/jhelvy/gh/0gw/P4A/2022-Spring/class/9-data-frames/data/data.csv"
```

## 2. Import the data

```
library(readr)
df <- read_csv(pathToData)
```

# Using the **here** package to make file paths

The `here()` function builds the path to your **root** to your *working directory* (this is where your `.Rproj` file lives!)

```
here()
```

```
#> [1] "/Users/jhelvy/gh/0gw/P4A/2022-Spring/class/9-data-frames"
```

The `here()` function builds the path to files *inside* your working directory

```
path_to_data <- here('data', 'data.csv')  
path_to_data
```

```
#> [1] "/Users/jhelvy/gh/0gw/P4A/2022-Spring/class/9-data-frames/data/data.csv"
```

# Avoid hard-coding file paths!

(they can break on different computers)

```
path_to_data <- 'data/data.csv'  
path_to_data
```

```
#> [1] "data/data.csv"
```



Use the **here** package  
to make file paths



Art by Allison Horst

# Back to reading in data

```
path_to_data <- here('data', 'data.csv')  
data <- read_csv(path_to_data)
```

**Important:** Use `read_csv()` instead of `read.csv()`

10:00

# Your turn

- 1) Use the `here()` and `read_csv()` functions to load the `data.csv` file that is in the `data` folder. Name the data frame object `df`.
- 2) Use the `df` object to answer the following questions:

- How many rows and columns are in the data frame?
- What type of data is each column?
- Preview the different columns - what do you think this data is about? What might one row represent?
- How many unique airports are in the data frame?
- What is the earliest and latest observation in the data frame?
- What is the lowest and highest cost of any one repair in the data frame?

# Next week: better data wrangling with **dplyr**



Art by Allison Horst

# Select rows with filter()

Example: Filter rows to find which Beatles members are still alive?

**Base R:**

```
beatles[beatles$deceased == FALSE,]
```

**dplyr:**

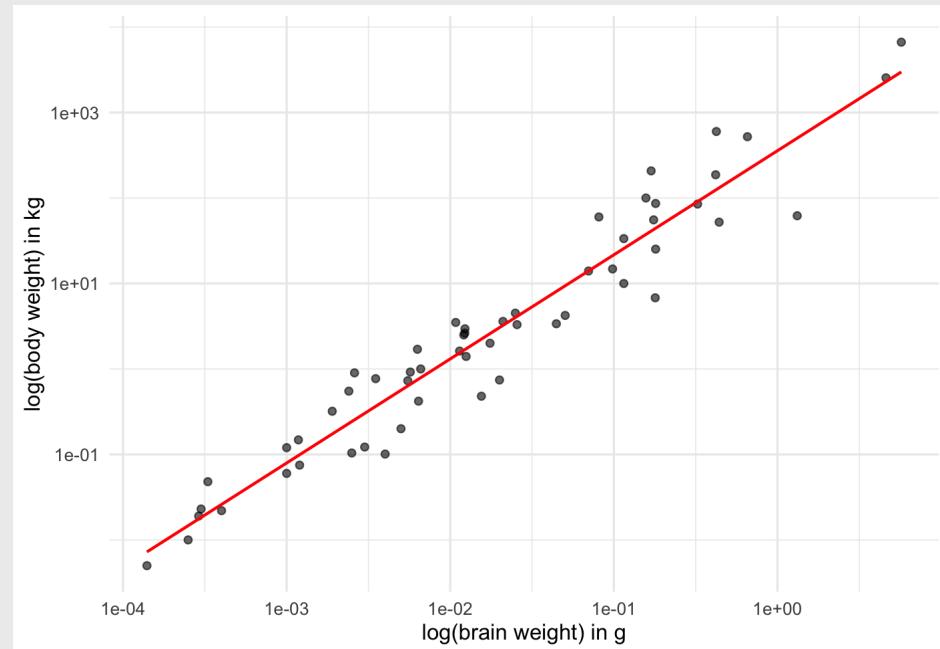
```
filter(beatles, deceased == FALSE)
```

# In 2 weeks: plotting with **ggplot2**

Translate data...

```
#> # A tibble: 11 × 2
#>   brainwt    bodywt
#>   <dbl>     <dbl>
#> 1 0.001      0.06
#> 2 0.0066     1
#> 3 0.00014    0.005
#> 4 0.0108     3.5
#> 5 0.0123     2.95
#> 6 0.0063     1.7
#> 7 4.60       2547
#> 8 0.0003     0.023
#> 9 0.655      521
#> 10 0.419     187
#> 11 0.0035    0.77
```

...into *information*



# A note about HW9

- You have what you need to start now.
- It will be *much* easier if you use the **dplyr** functions (i.e. read ahead).