



Week 2: *Data Wrangling*

 EMSE 6035: Marketing Analytics for Design Decisions

 John Paul Helveston

 September 04, 2024

Required Packages (check `practice.R` file)

Make sure you have these libraries installed:

```
install.packages(c("tidyverse", "here"))
```

Remember: you only need to install packages once!

Once installed, you'll need to *load* the libraries every time you open RStudio:

```
library(tidyverse)  
library(here)
```

Week 2: *Data Wrangling*

1. Working with data frames
2. Data wrangling with the *tidyverse*

BREAK

3. Project proposals

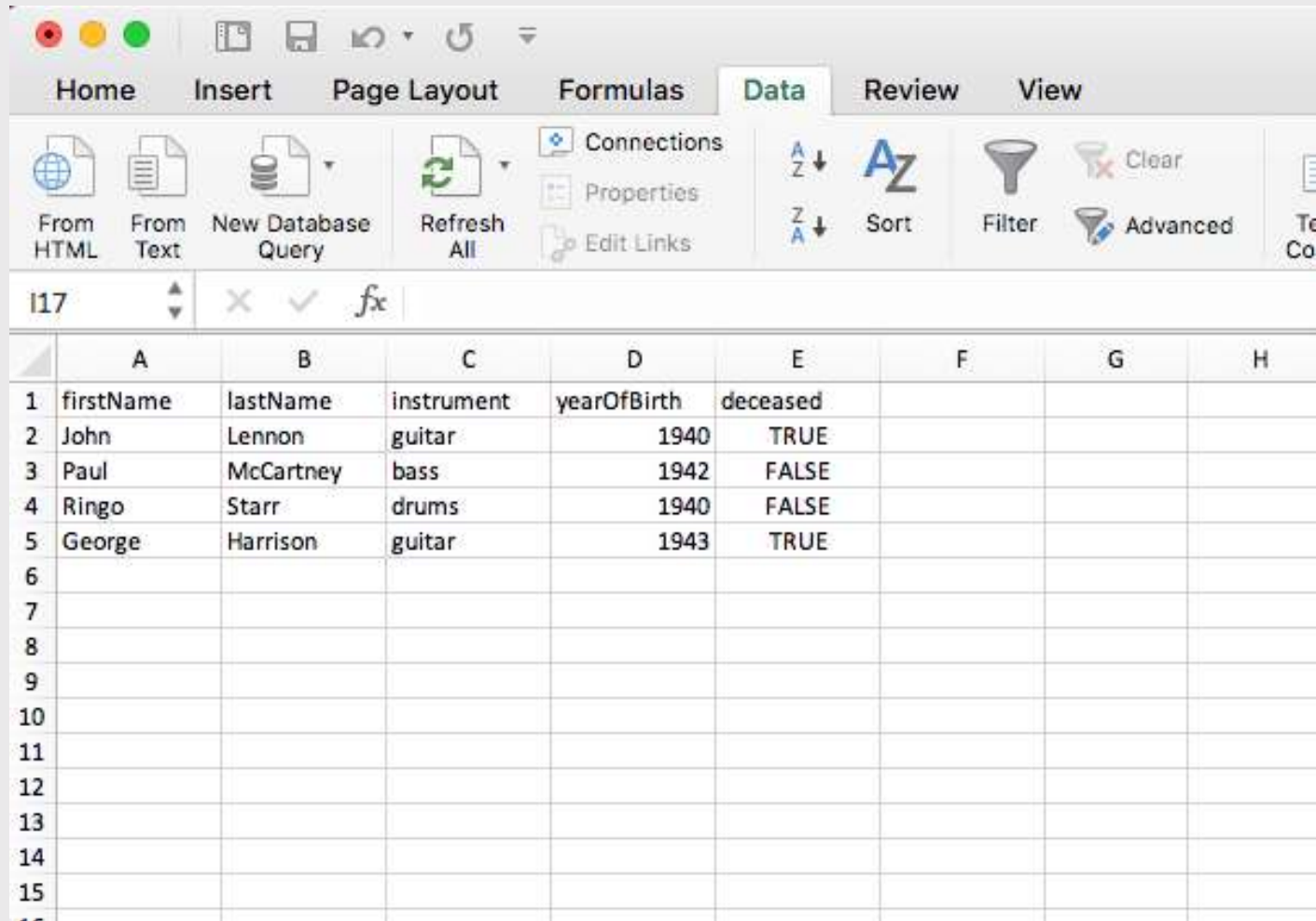
Week 2: *Data Wrangling*

1. Working with data frames
2. Data wrangling with the *tidyverse*

BREAK

3. Project proposals

The data frame...in Excel



The screenshot shows the Microsoft Excel interface with the 'Data' tab selected. The ribbon includes options for 'From HTML', 'From Text', 'New Database Query', 'Refresh All', 'Connections', 'Properties', 'Edit Links', 'Sort', 'Filter', 'Advanced', and 'Clear'. The active cell is I17. The data frame is located in the range A1:E5.

	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
```

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
```


Take a look at the `beatles` data frame in `practice.R`

Getting data into R

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

NOTE: 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 more about a package data set:

```
?msleep
```

Back to `practice.R`

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

```
#> [1] "/Users/jhelvy/gh/teaching/MADD/2024-Fall/class/2-data-wrangling/data/data.csv"
```

2. Import the data

```
library(tidyverse)  
data <- read_csv(path_to_data)
```

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/teaching/MADD/2024-Fall/class/2-data-wrangling"
```

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/teaching/MADD/2024-Fall/class/2-data-wrangling/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"
```



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()`

Your turn

10:00

- 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 `data`.
- 2) Use the `data` object to answer the following questions:
 - How many rows and columns are in the data frame?
 - What type of data is each column? (Just look, don't need to type out the answer)
 - 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?

Week 2: *Data Wrangling*

1. Working with data frames

2. **Data wrangling with the *tidyverse***

BREAK

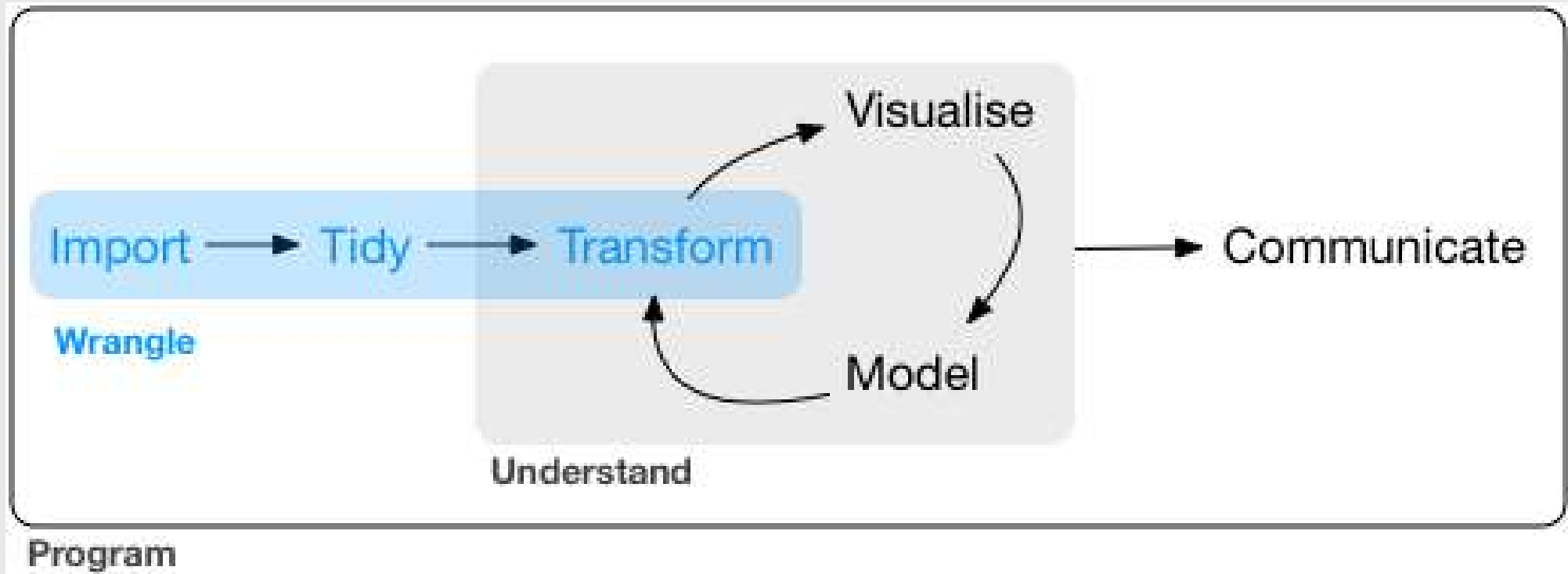
3. Project proposals

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



Art by [Allison Horst](#)

80% of the job is data wrangling



Today: data wrangling with **dplyr**



Art by [Allison Horst](#)

The main `dplyr` "verbs"

"Verb"	What it does
<code>select()</code>	Select columns by name
<code>filter()</code>	Keep rows that match criteria
<code>arrange()</code>	Sort rows based on column(s)
<code>mutate()</code>	Create new columns
<code>summarize()</code>	Create summary values

Core `tidyverse` concept: **Chain functions together with "pipes"**

`%>%`

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

```
data %>%  
  do_something() %>%  
  do_something_else()
```


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
```

Select columns with `select()`

Subset Variables (Columns)



Select columns with `select()`

```
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
```

Select columns with `select()`

Select the columns `firstName` & `lastName`

```
beatles %>%  
  select(firstName, lastName)
```

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

Select columns with `select()`

Use the `-` sign to drop columns

```
beatles %>%  
  select(-firstName, -lastName)
```

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

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()`

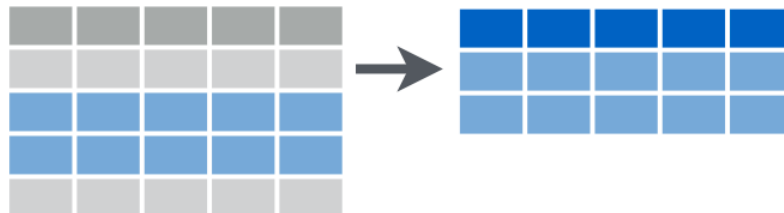
Select the columns that end with `"Name"`:

```
beatles %>%  
  select(ends_with("Name"))
```

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

Keep specific rows with `filter()`

Subset Observations (Rows)



Keep specific rows with `filter()`

Keep only the rows with band members born after 1941

```
#> # 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
```

Keep specific rows with `filter()`

Keep only the rows with band members born after 1941

```
beatles %>%  
  filter(yearOfBirth > 1941)
```

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

Keep specific rows with `filter()`

Keep only the rows with band members born after 1941 **& are still living**

```
beatles %>%  
  filter(yearOfBirth > 1941, deceased == FALSE)
```

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

```
beatles %>%  
  filter((yearOfBirth > 1941) & (deceased == FALSE))
```

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

```
data %>%  
  filter(!is.na(variable))
```

Combine `filter()` and `select()`

Get the **first & last name** of members born after 1941 & are still living

```
beatles %>%  
  filter(yearOfBirth > 1941, deceased == FALSE) %>%  
  select(firstName, lastName)
```

```
#> # A tibble: 1 × 2  
#>   firstName lastName  
#>   <chr>      <chr>  
#> 1 Paul      McCartney
```

Your turn

10:00

- 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 `data`.
- 2) Use the `data` object and the `select()` and `filter()` functions to answer the following questions:
 - Create a new data frame, `dc`, that contains only the rows from DC airports.
 - Create a new data frame, `dc_dawn`, that contains only the rows from DC airports that occurred at dawn.
 - Create a new data frame, `dc_dawn_birds`, that contains only the rows from DC airports that occurred at dawn and only the columns about the *species* of bird.
 - How many unique species of birds have been involved in accidents at DC airports?

Create new variables with `mutate()`

Make New Variables





Create new variables with `mutate()`

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

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

```
#> # A tibble: 4 × 6  
#>   firstName lastName instrument yearOfBirth deceased   age  
#>   <chr>      <chr>      <chr>          <dbl> <lgl>    <dbl>  
#> 1 John      Lennon      guitar         1940 TRUE      82  
#> 2 Paul      McCartney  bass           1942 FALSE     80  
#> 3 Ringo     Starr      drums           1940 FALSE     82  
#> 4 George    Harrison   guitar          1943 TRUE      79
```

You can *immediately* use new variables

```
beatles %>%  
  mutate(  
    age = 2022 - yearOfBirth,  
    meanAge = mean(age))
```

```
#> # A tibble: 4 × 7  
#>   firstName lastName instrument yearOfBirth deceased   age meanAge  
#>   <chr>      <chr>      <chr>          <dbl> <lgl>    <dbl>   <dbl>  
#> 1 John      Lennon      guitar        1940  TRUE     82     80.8  
#> 2 Paul      McCartney  bass          1942  FALSE    80     80.8  
#> 3 Ringo     Starr      drums         1940  FALSE    82     80.8  
#> 4 George    Harrison  guitar        1943  TRUE     79     80.8
```

Handling if/else conditions

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

```
beatles %>%  
  mutate(playsGuitar = ifelse(instrument == "guitar", TRUE, FALSE))
```

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

Sort data frame with `arrange()`

Sort `beatles` data frame by year of birth

```
beatles %>%  
  arrange(yearOfBirth)
```

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

Sort data frame with `arrange()`

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

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

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

Sort rows with `arrange()`

Compute the band member age, then sort based on the youngest:

```
beatles %>%  
  mutate(age = 2022 - yearOfBirth) %>%  
  arrange(age)
```

```
#> # A tibble: 4 × 6  
#>   firstName lastName instrument yearOfBirth deceased age  
#>   <chr>      <chr>      <chr>      <dbl> <lgl>      <dbl>  
#> 1 George    Harrison guitar      1943 TRUE       79  
#> 2 Paul      McCartney bass        1942 FALSE      80  
#> 3 John      Lennon guitar      1940 TRUE       82  
#> 4 Ringo     Starr drums        1940 FALSE      82
```

Your turn

10:00

- 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 `data`.
- 2) Using the `data` object, 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.
- 3) Remove rows that have `NA` for `cost_repairs_infl_adj` and re-arrange the resulting data frame based on the highest height and most expensive cost

Break

05 : 00

Week 2: *Data Wrangling*

1. Working with data frames
2. Data wrangling with the *tidyverse*

BREAK

3. Project proposals

Project Proposal Guidelines

Proposal Items

Item	Description
Abstract	Product / technology in just a few sentences
Introduction	Description, picture, background
Market Opportunity	Identify your customer, competitors, and market size
Product Attributes	2-4 key variables related to product's design and performance
Research Questions	2-4 research questions you hope to answer about your product
Questions	Major outstanding questions to be resolved

Today

Market Opportunity

- Identify customer
- Identify competitors
- Identify market size

Product Attributes

Features your *customer* cares about

Research Questions

Decisions are trying to inform

Example: **Folding solar panels**



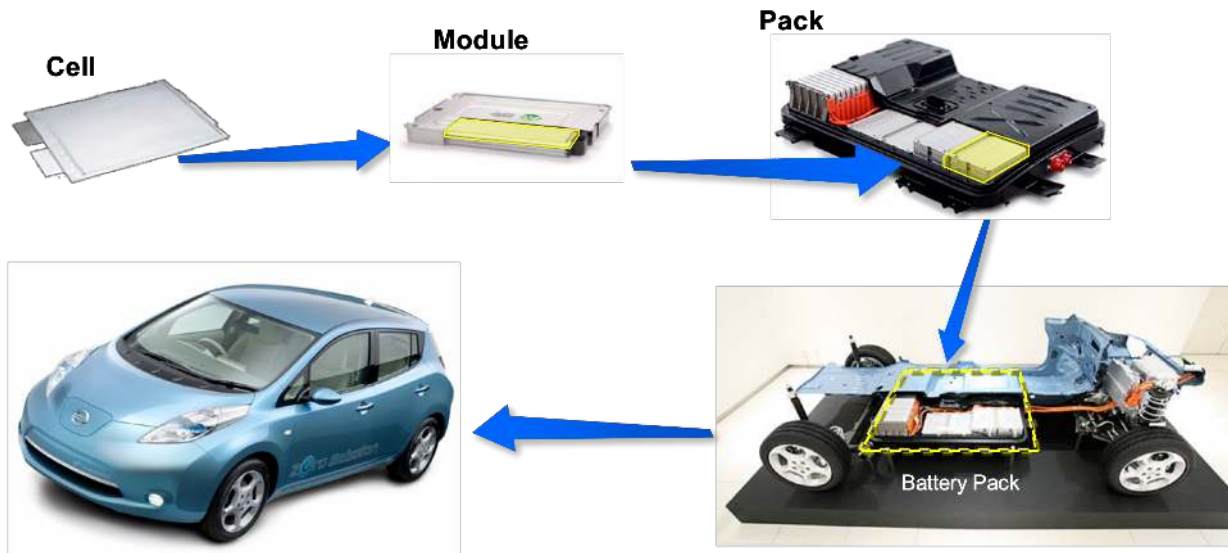
Who is your customer?

- General public?
- Outdoor enthusiasts?
- Emergency gear?

Competitors?

- Similar folding panels
- Batteries?

Example: **Electric vehicle battery**



Who is your customer?

- Car buyers

Competitors?

- Hybrid vehicles?
- Efficient gasoline vehicles?

Product Attributes

Features your *customer* cares about

Research Questions

Decisions you are trying to inform

Product Diagram



Durability

Degradation Protections

Attribute Units

Price – USD

Weight – Kg

Power Output – Watts

Durability – Months

Portability – LxWxH

Decision Variable Units

Power Density – W/Kg

Degradation Rate – Hours

Packing Design – Cm^3



Portability

Power Output

Power Density

Packing Design



Weight

Image Sources:
1. <https://www.deviantart.com/kota321/art/Ramp-Sun-244479413>
2. <https://www.pennstate.com/product/intel-certified-tools/fabric-printing>
3. <https://www.thiswayimback.com/2015/05/10/foldable-solar-panel-charger/>

Product Attributes Table (example)

	Product:	Pocket Charge				
	Description:	Flexible, foldable, portable solar charger				
		Features			Competitors	
		Range	Units	Demand	Aims Solar Panel	SUAOKI Solar Charger
Product Attributes	Price	\$60 - \$225	USD / kW	-	225	160
	Weight	1 - 3	kg	-	2.6	2.06
	Power Output	100 - 500	W	+	120	60
	Durability	12 - 60	months	+	60	12
	Portability	200 - 2800	cm^3	+	20.6"x11"x 1.2"	11.5"x7.1"x2.9"

Team Proposals

1. Re-arrange tables to sit with your team
2. Discuss & identify your customer & potential competitors
3. Discuss & identify key *Product Attributes & Research Questions*
4. Start building out your model relationships table (copy from [this example](#))

Suggestions

- You may want to start with simple bullet lists
- Start with more items rather than fewer (can always cut back later)