TODAY'S CLASS

Housekeeping:

- ❑ Midterm Project due by 11PM on Sunday, 7th Nov. 2021

6:00PM – 6:50PM: **dplyr** data manipulation

7:00PM – 7:45PM: **dplyr** data manipulation

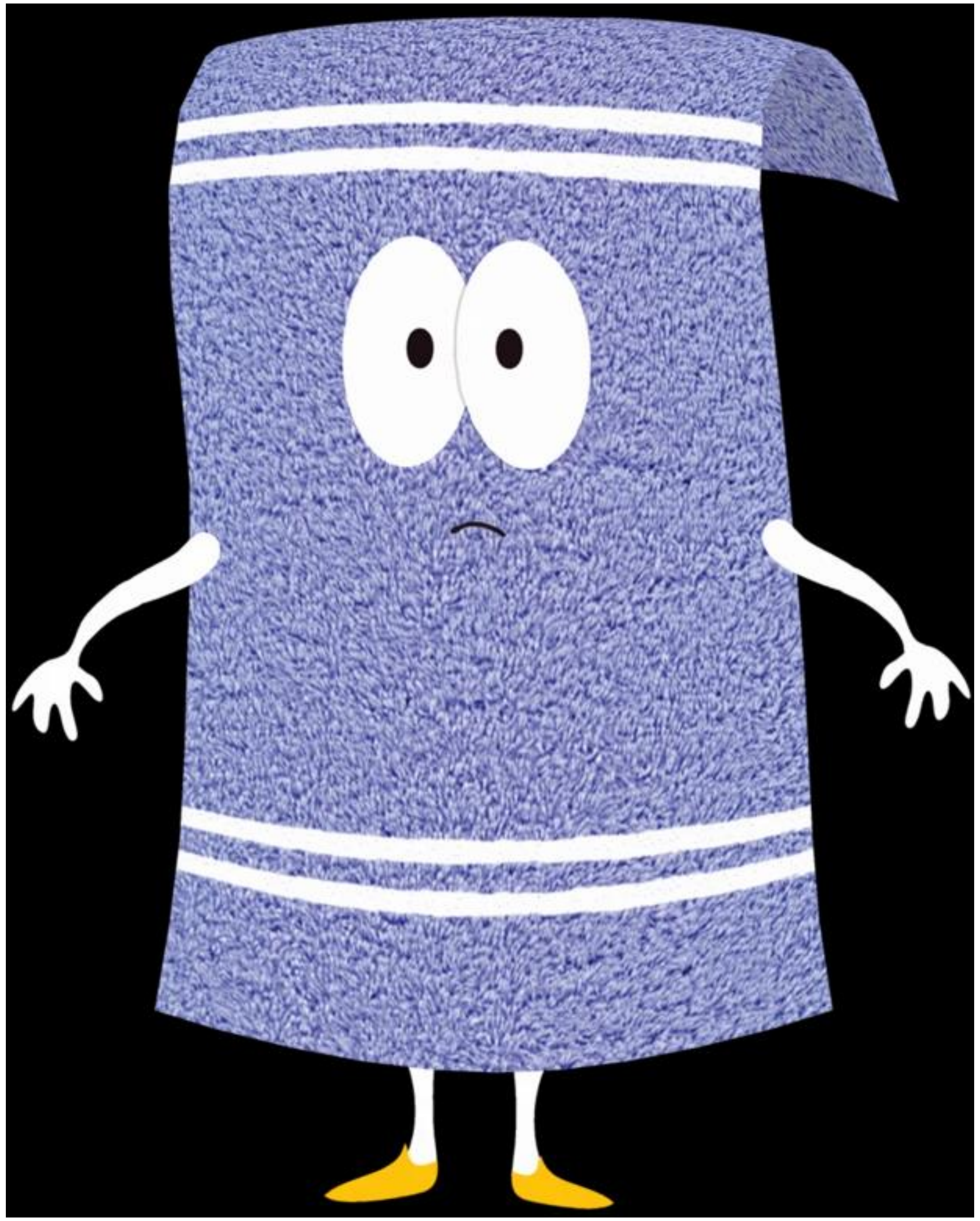
8:00PM – 8:50PM: **tidyr** functions for data reshaping

9:00PM – 9:50PM: MBTA group challenge

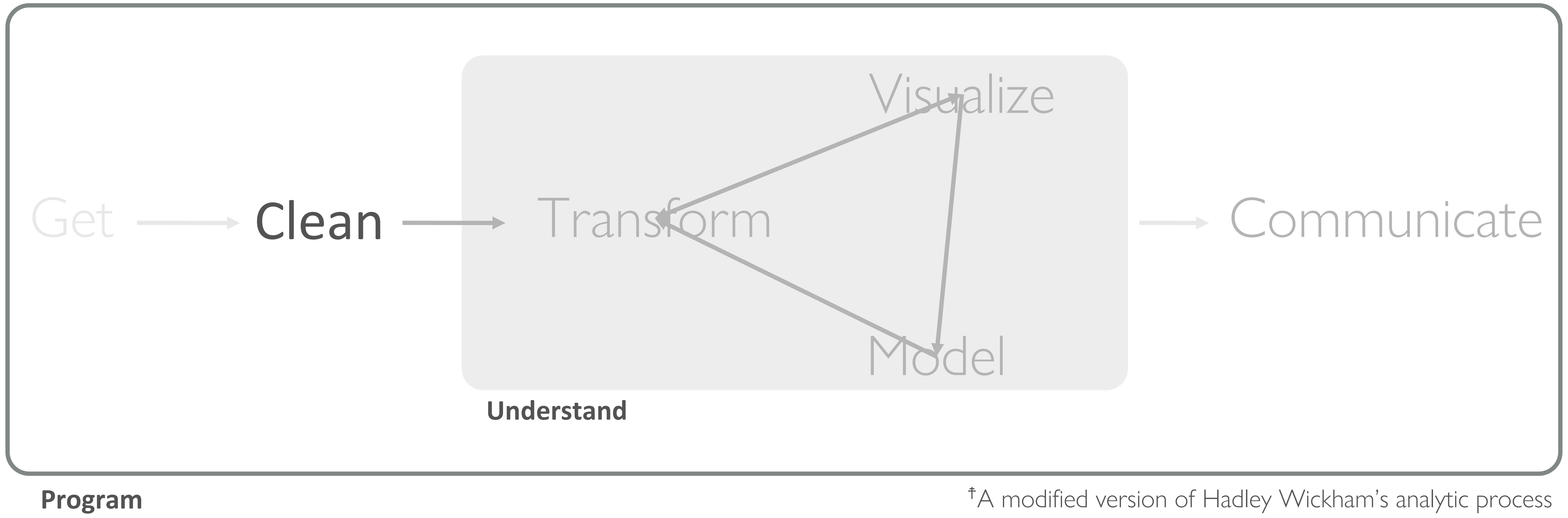
DON'T FORGET!

- Go to the course website to download today's Week 3 material.

- Leverage the .R scripts so you don't have to type everything!
  - ✓ Take notes by commenting in the scripts!

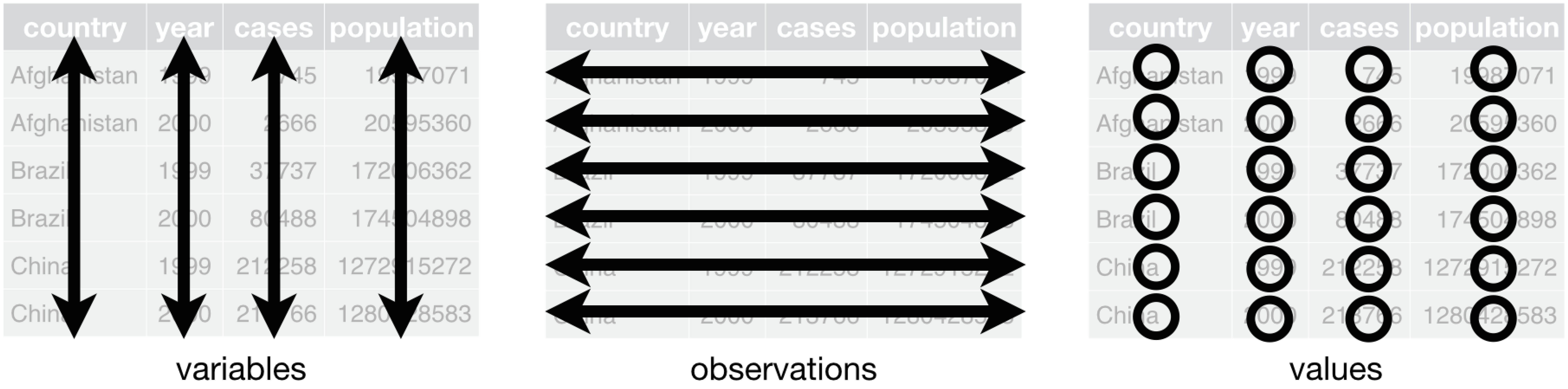


CLEAN & TIDY DATA STRUCTURES



†A modified version of Hadley Wickham's analytic process

WHAT IS TIDY DATA?

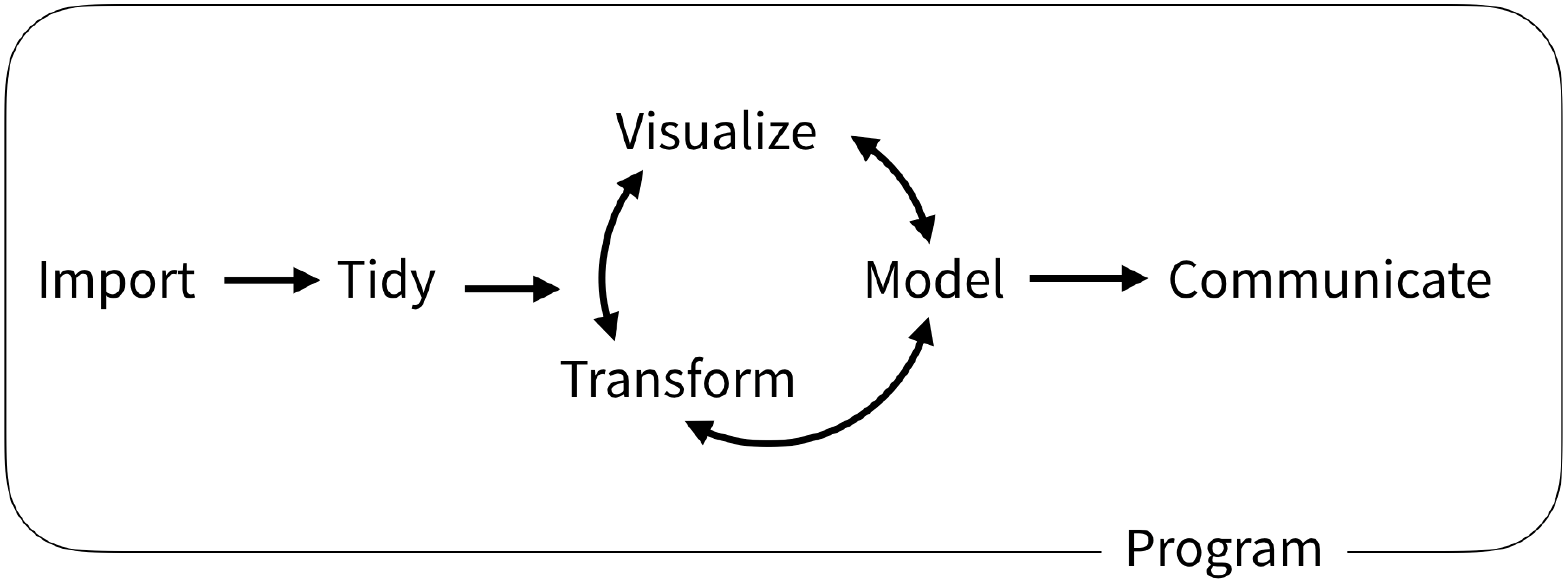


WHERE ARE THE VARIABLES, OBSERVATIONS, AND VALUES?

country	year	cases	population
Afghanistan	1999	745	19987071
Afghanistan	2000	2666	20595360
Brazil	1999	37737	172006362
Brazil	2000	80488	174504898
China	1999	212258	1272915272
China	2000	213766	1280428583

INTRO TO THE TIDYVERSE

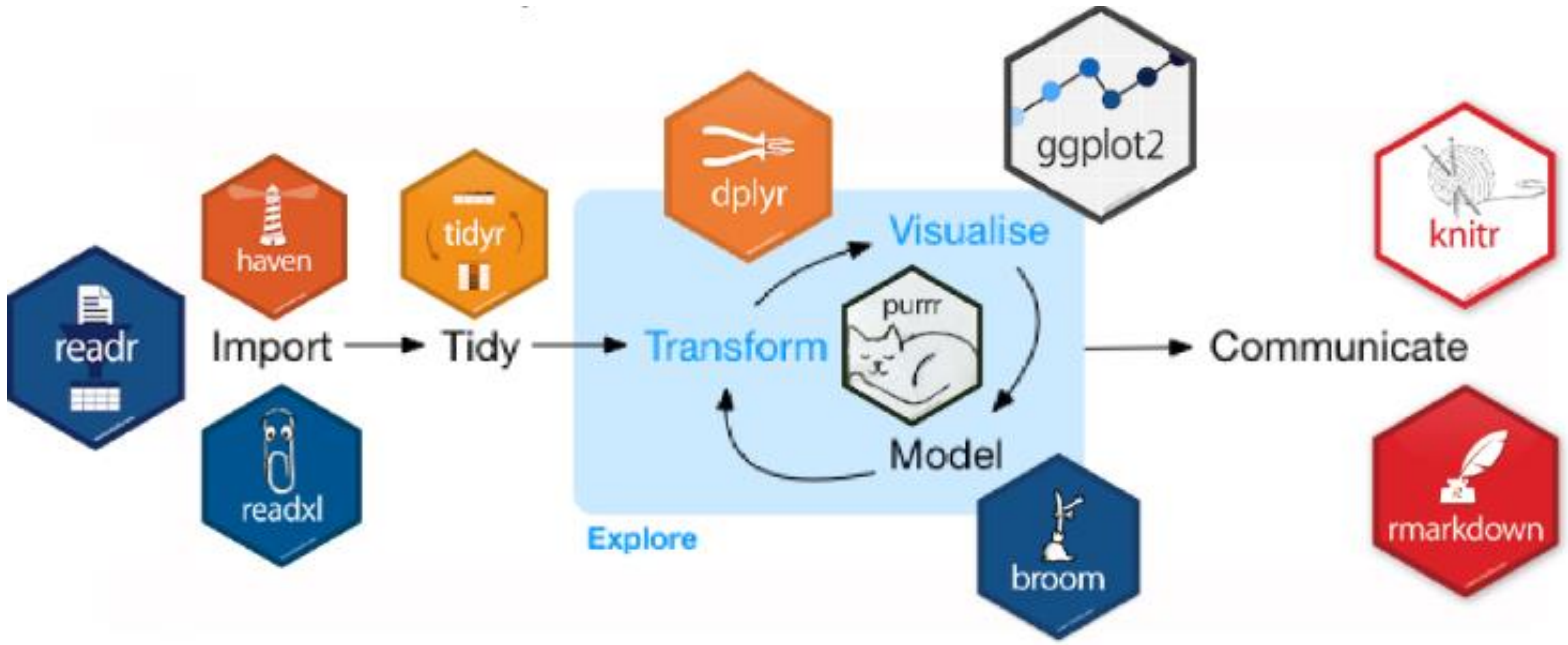
TYPICAL DATA SCIENCE WORKFLOW



WHAT IS THE TIDYVERSE?

An opinionated collection of packages...

designed to simplify data analysis.



INSTALLING/LOADING CORE TIDYVERSE PACKAGES

install.packages("tidyverse")

does the equivalent of...

install.packages("dplyr")
install.packages("ggplot2")
install.packages("tidyr")
install.packages("tibble")
install.packages("readr")
install.packages("purrr")
install.packages("stringr")
install.packages("forcats")

library(tidyverse)

does the equivalent of...

library(dplyr)
library(ggplot2)
library(tidyr)
library(tibble)
library(readr)
library(purrr)
library(stringr)
library(forcats)

OTHER TIDYVERSE PACKAGES NOT AUTOMATICALLY LOADED

The Tidyverse also includes many other packages that are not automatically loaded with `library(tidyverse)`.

Use `library()` to load each package and leverage its more specialized capabilities.

```
# load core tidyverse packages
library(tidyverse)

# load other non-core tidyverse packages
# these aren't the only non-core tidyverse packages
library(readxl)     # reading excel spreadsheets
library(lubridate)  # working with dates and datetimes
library(magrittr)   # additional pipe operators
library(glue)       # an alternative to paste
```

pipe operator

Chaining functions together with the pipe operator

%>%

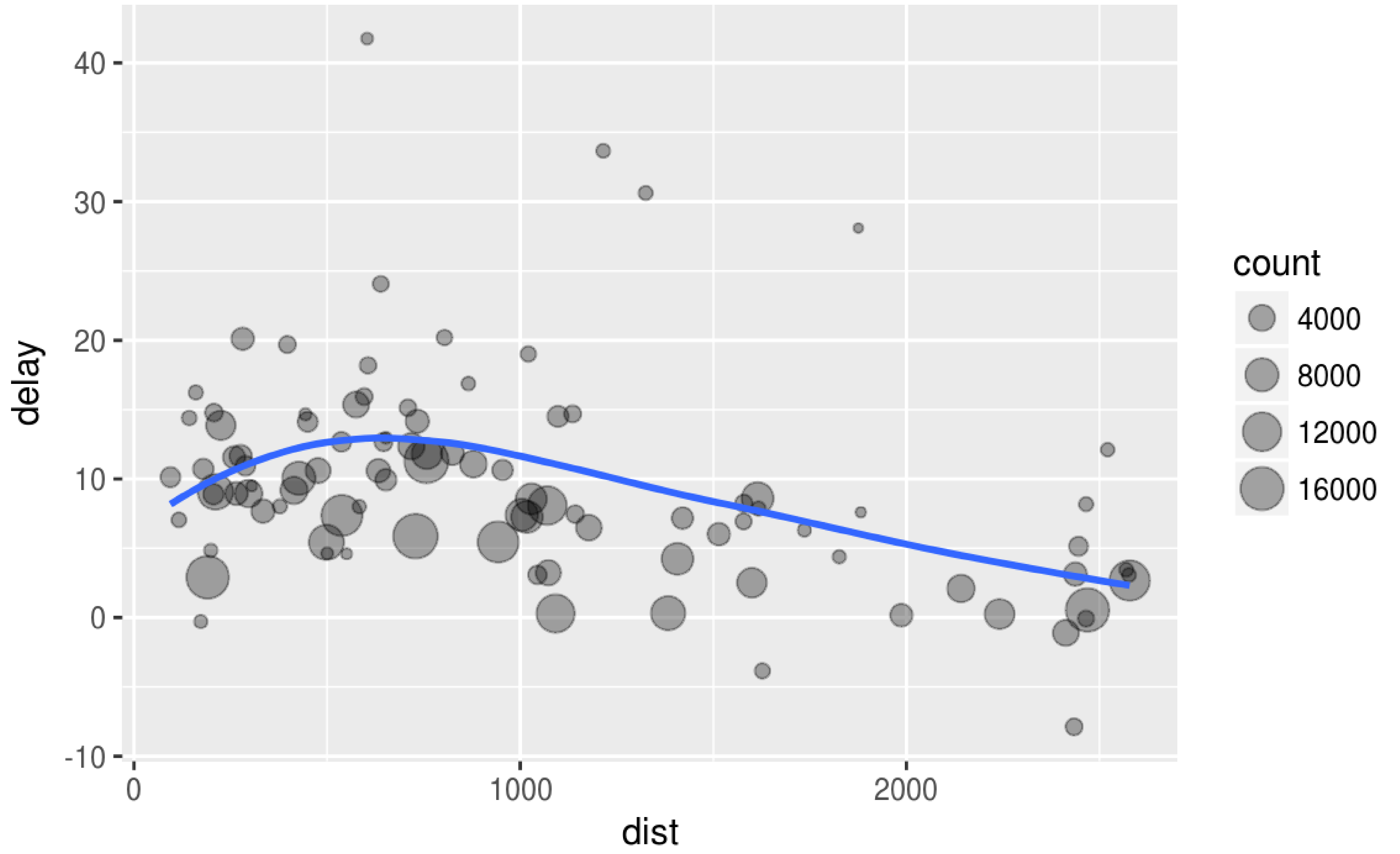
STREAMLINING OUR ANALYSIS

Imagine that we want to explore the relationship between the distance and average delay for each location.

Using what you know about **dplyr** and **ggplot**, you might write code like this:

```
by_dest <- group_by(flights, dest)
delay <- summarise(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != “HNL")

ggplot(data = delay, mapping = aes(x = dist, y = delay)) +
  geom_point(aes(size = count), alpha = 1/3) +
  geom_smooth(se = FALSE)
```



STREAMLINING OUR ANALYSIS

Imagine that we want to explore the relationship between the distance and average delay for each location.

Using what you know about dplyr and ggplot, you might write code like this:

```
by_dest <- group_by(flights, dest)
delay <- summarise(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != “HNL")

ggplot(data = delay, mapping = aes(x = dist, y = delay)) +
  geom_point(aes(size = count), alpha = 1/3) +
  geom_smooth(se = FALSE)
```

This code does four things:

1. ?
2. ?
3. ?
4. ?

STREAMLINING OUR ANALYSIS

Imagine that we want to explore the relationship between the distance and average delay for each location.

Using what you know about dplyr and ggplot, you might write code like this:

```
by_dest <- group_by(flights, dest)
delay <- summarise(by_dest,
  count = n(),
  dist = mean(distance, na.rm = TRUE),
  delay = mean(arr_delay, na.rm = TRUE)
)
delay <- filter(delay, count > 20, dest != “HNL")

ggplot(data = delay, mapping = aes(x = dist, y = delay)) +
  geom_point(aes(size = count), alpha = 1/3) +
  geom_smooth(se = FALSE)
```

This code does four things:

1. grouping by destination
2. summarizing count, distance, and delay
3. filtering out low counts and Honolulu
4. creating a plot

STREAMLINING OUR ANALYSIS

We can streamline our code to make it more ***efficient*** and ***legible***

```
library(dplyr)
x <- 1:15
sum(x)
x %>% sum()
```

These do the same thing

Try it!

%>%

x sum(_)

STREAMLINING OUR ANALYSIS

Let's re-write our code using the pipe **(%>%)** operator:

```
flights %>%
    group_by(dest) %>%
    summarise(count = n(),
        dist = mean(distance, na.rm = TRUE),
        delay = mean(arr_delay, na.rm = TRUE)) %>%
    filter(count > 20, dest != "HNL") %>%
    ggplot(aes(x = dist, y = delay)) +
      geom_point(aes(size = count), alpha = 1/3) +
      geom_smooth(se = FALSE)
```

This code does four things in a very *efficient* & *readable* manner:

1. grouping by destination
2. summarizing count, distance, and delay
3. filtering out low counts and Honolulu
4. creating a plot

STREAMLINING OUR ANALYSIS

Let's re-write our code using the pipe **(%>%)** operator:

```
flights %>%
    group_by(dest) %>%
    summarise(count = n(),
        dist = mean(distance, na.rm = TRUE),
        delay = mean(arr_delay, na.rm = TRUE)) %>%
    filter(count > 20, dest != "HNL") %>%
    ggplot(aes(x = dist, y = delay)) +
      geom_point(aes(size = count), alpha = 1/3) +
      geom_smooth(se = FALSE)
```

COMMUNICATE WITH THE TIDYVERSE

Start with single year

Called the pipe; pronounced "then"

Called the reverse assignment operator; pronounced "creates"

```
gapminder %>%
  filter(year == 2015) ->
  gapminder15
```

dplyr

Phonics are important!

```
gapminder %>%
  filter(year == 2015) ->
  gapminder15
```

Take the gapminder data, then

filter rows where year equals 2015, creating

gapminder15 variable

- Source from Hadley Wickham, February 2019: https://speakerdeck.com/hadley/welcome-to-the-tidyverse
- Reverse assignment operator

PIPE OPERATOR KEYBOARD SHORTCUT

Ctrl + Shift + M

TIDYING DATA

tidyr

You learned four key tidyr functions that allow you to solve the vast majority of your data tidying challenges:

- gather:
- spread:
- separate:
- unite:

tidyr

You learned four key tidyr functions that allow you to solve the vast majority of your data tidying challenges:

- **gather:** transforms data from wide to long
- **spread:** transforms data from long to wide
- **separate:** splits a single column into multiple columns
- **unite:** combines multiple columns into a single column

WIDE VS. LONG DATA

wide

id	x	y	z
1	a	c	e
2	b	d	f

long

id	key	val
1	x	a
2	x	b
1	y	c
2	y	d
1	z	e
2	z	f

PREREQUISITES

PREREQUISITES

- Make sure your working directory is set to the course folder
- Should have downloaded the folder for today’s class from the course website
- We will use the various data sets in the data folder

PACKAGE PREREQUISITE

```
library(tidyverse)
#> Loading tidyverse: ggplot2
#> Loading tidyverse: tibble
#> Loading tidyverse: tidyr
#> Loading tidyverse: readr
#> Loading tidyverse: purrr
#> Loading tidyverse: dplyr
#> Conflicts with tidy packages ----------------------------------------------
#> filter(): dplyr, stats
#> lag():    dplyr, stats
```

gather()

Transform data from wide to long

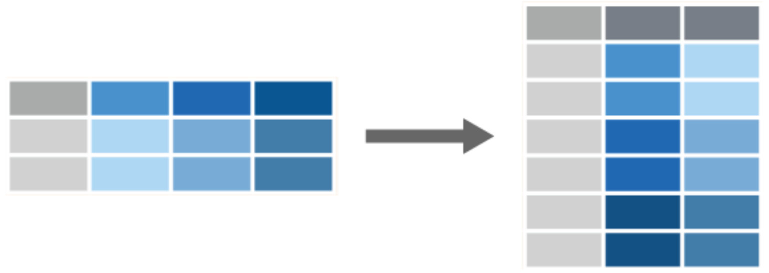
gather()

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% gather(Year, n, 2:4)
```

dataframe to reshape

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000



gather()

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% gather(Year, n, 2:4)
```

name of the new "key" column

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

gather()

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% gather(Year, n, 2:4)
```

name of the new “value” column

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

gather()

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

key

value

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

```
cases %>% gather(Year, n, 2:4)
```

columns to collapse

gather()

Code alternatives:

```
# These all produce the same results:
    cases %>% gather(Year, n, `2011`:`2013`)
    cases %>% gather(Year, n, `2011`, `2012`, `2013`)
    cases %>% gather(Year, n, 2:4)
    cases %>% gather(Year, n, -Country)

# Also note that if you do not supply arguments for na.rm or convert values then the defaults are used
```

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

gather()

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

YOUR TURN!

1. *Import the* bomber_wide.rds *file in the data folder*
2. *Reshape this data from wide to long, but keep the Type and MD columns as-is*

SOLUTION

```
read_rds("data/bomber_wide.rds") %>%
  gather(Year, Value, -c(Type, MD))
    Type   MD Year Value
1  Bomber  B-1 1996 26914
2  Bomber  B-2 1996  2364
3  Bomber B-52 1996 28511
4  Bomber  B-1 1997 25219
5  Bomber  B-2 1997  2776
6  Bomber B-52 1997 26034
7  Bomber  B-1 1998 24205
8  Bomber  B-2 1998  2166
9  Bomber B-52 1998 25639
10 Bomber  B-1 1999 23306
11 Bomber  B-2 1999  3672
12 Bomber B-52 1999 24500
13 Bomber  B-1 2000 25013
14 Bomber  B-2 2000  4543
```

spread()

Transform data from long to wide

spread()

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% spread(Year, n)
```

dataframe
to reshape

spread()

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% spread(Year, n)
```

column to use as keys
(new column names)

spread()

Country	Year	n
FR	2011	7000
DE	2011	5800
US	2011	15000
FR	2012	6900
DE	2012	6000
US	2012	14000
FR	2013	7000
DE	2013	6200
US	2013	13000

Country	2011	2012	2013
FR	7000	6900	7000
DE	5800	6000	6200
US	15000	14000	13000

```
cases %>% spread(Year, n)
```

column to use as values
(new column cells)

YOUR TURN!

1. *Import the* bomber_long.rds *file in the data folder*

2. *Reshape this data from long to wide*

Type, MD, FY, Cost, FH, Gallons

SOLUTION

```
read_rds("data/bomber_long.rds") %>%
  spread(Output, Value)
    Type   MD   FY      Cost    FH   Gallons
1  Bomber  B-1 1996  72753781 26914  88594449
2  Bomber  B-1 1997  71297263 25219  85484074
3  Bomber  B-1 1998  84026805 24205  85259038
4  Bomber  B-1 1999  71848336 23306  79323816
5  Bomber  B-1 2000  58439777 25013  86230284
6  Bomber  B-1 2001  94946077 25059  86892432
7  Bomber  B-1 2002  96458536 26581  89198262
8  Bomber  B-1 2003  68650070 21491  74485788
9  Bomber  B-1 2004 101895634 28118 101397707
10 Bomber  B-1 2005 124816690 21859  78410415
11 Bomber  B-1 2006 174627869 20163  69984142
12 Bomber  B-1 2007 204486404 24629  85112485
13 Bomber  B-1 2008 266109848 23024  78084791
14 Bomber  B-1 2009 185902082 23065  81030579
15 Bomber  B-1 2010 237413270 23398  81253214
```

separate()

Split a single column into multiple columns

separate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% separate(date, c("year", "month", "day"), sep = "-")
```

dataframe
to reshape

separate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% separate(date, c(”year”, “month”, “day”), sep = “-”)
```

column to split into multiple columns

separate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% separate(date, c(”year”, “month”, “day”), sep = “-”)
```

names of the new variable columns

separate()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% separate(date, c("year", "month", "day"), sep = "-")
```

how to separate current variable

separate()

Code alternatives:

```
# These all produce the same results:
        storms %>% separate(date, c("year", "month", “day"))
        storms %>% separate(date, c("year", "month", "day"), sep = "-")

# By default, if no separator is specified, will separate by any regular expression that matches          # any sequence of non-alphanumeric
values
```

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

separate()

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

YOUR TURN!

1. *Import the* bomber_combined.rds *file in the data folder*

2. *Separate the AC variable into "Type" and "MD". Hint: An example of a Type is "Bomber".*

SOLUTION

```
read_rds("data/bomber_combined.rds") %>%
  separate(AC, into = c("Type", "MD"), sep = " ")
    Type   MD   FY      Cost    FH   Gallons
1  Bomber  B-1 1996  72753781 26914  88594449
2  Bomber  B-1 1997  71297263 25219  85484074
3  Bomber  B-1 1998  84026805 24205  85259038
4  Bomber  B-1 1999  71848336 23306  79323816
5  Bomber  B-1 2000  58439777 25013  86230284
6  Bomber  B-1 2001  94946077 25059  86892432
7  Bomber  B-1 2002  96458536 26581  89198262
8  Bomber  B-1 2003  68650070 21491  74485788
9  Bomber  B-1 2004 101895634 28118 101397707
10 Bomber  B-1 2005 124816690 21859  78410415
11 Bomber  B-1 2006 174627869 20163  69984142
12 Bomber  B-1 2007 204486404 24629  85112485
13 Bomber  B-1 2008 266109848 23024  78084791
14 Bomber  B-1 2009 185902082 23065  81030579
```

unite()

Combine multiple columns into a single column

unite()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% unite(date, year, month, day, sep = "-")
```

dataframe to reshape

unite()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% unite(date, year, month, day, sep = “-”)
```

name of new “merged” column

unite()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% unite(date, year, month, day, sep = "-")
```

columns to merge

unite()

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

```
storms %>% unite(date, year, month, day, sep = “-”)
```

separator to use
btwn merged values

unite()

Code alternatives:

```
# These all produce the same results:
    storms %>% unite(date, year, month, day, sep = “_")
    storms %>% unite(date, year, month, day)

# If no separator is identified, "_" will automatically be used
```

storm	wind	pressure	date
Alberto	110	1007	2000-08-12
Alex	45	1009	1998-07-30
Allison	65	1005	1995-06-04
Ana	40	1013	1997-07-01
Arlene	50	1010	1999-06-13
Arthur	45	1010	1996-06-21

storm	wind	pressure	year	month	day
Alberto	110	1007	2000	08	12
Alex	45	1009	1998	07	30
Allison	65	1005	1995	06	04
Ana	40	1013	1997	07	1
Arlene	50	1010	1999	06	13
Arthur	45	1010	1996	06	21

YOUR TURN!

1. *Import the* bomber_prefix.rds *file in the data folder*
2. *Unite the prefix and number columns into a "MD" variable with "-" separator*

SOLUTION

```
read_rds("data/bomber_prefix.rds") %>%
  unite(MD, prefix, number, sep = "-")
     Type   MD   FY  Output     Value
1   Bomber  B-1 1996     FH     26914
2   Bomber  B-1 1997     FH     25219
3   Bomber  B-1 1998     FH     24205
4   Bomber  B-1 1999     FH     23306
5   Bomber  B-1 2000     FH     25013
6   Bomber  B-1 2001     FH     25059
7   Bomber  B-1 2002     FH     26581
8   Bomber  B-1 2003     FH     21491
9   Bomber  B-1 2004     FH     28118
10  Bomber  B-1 2005     FH     21859
11  Bomber  B-1 2006     FH     20163
12  Bomber  B-1 2007     FH     24629
13  Bomber  B-1 2008     FH     23024
14  Bomber  B-1 2009     FH     23065
```

CHALLENGE

1. *Import the* bomber_mess.rds *file in the data folder*

2. *Clean this data up so it looks like:*

```
# A tibble: 57 × 6
    Type   MD   FY     Cost    FH  Gallons
*   <chr> <chr> <chr>     <int> <int>     <int>
1  Bomber  B-1  1996  72753781 26914  88594449
2  Bomber  B-1  1997  71297263 25219  85484074
3  Bomber  B-1  1998  84026805 24205  85259038
4  Bomber  B-1  1999  71848336 23306  79323816
5  Bomber  B-1  2000  58439777 25013  86230284
6  Bomber  B-1  2001  94946077 25059  86892432
7  Bomber  B-1  2002  96458536 26581  89198262
8  Bomber  B-1  2003  68650070 21491  74485788
9  Bomber  B-1  2004 101895634 28118 101397707
10 Bomber  B-1  2005 124816690 21859  78410415
# ... with 47 more rows
```

SOLUTION

```
read_rds("data/bomber_mess.rds") %>%
  unite(col = MD, prefix:number, sep = "-") %>%
  separate(Metric, into = c("FY", "Output")) %>%
  spread(Output, Value) %>%
  as_tibble()
# A tibble: 57 × 6
    Type    MD    FY      Cost    FH   Gallons
*   <chr> <chr> <chr>     <int> <int>     <int>
1  Bomber   B-1  1996  72753781 26914  88594449
2  Bomber   B-1  1997  71297263 25219  85484074
3  Bomber   B-1  1998  84026805 24205  85259038
4  Bomber   B-1  1999  71848336 23306  79323816
5  Bomber   B-1  2000  58439777 25013  86230284
6  Bomber   B-1  2001  94946077 25059  86892432
7  Bomber   B-1  2002  96458536 26581  89198262
8  Bomber   B-1  2003  68650070 21491  74485788
9  Bomber   B-1  2004 101895634 28118 101397707
```

PRACTICE MAKES (NEARLY) PERFECT!

Let's review/apply what you've learned with a use-case

MBTA RIDERSHIP

GROUP WORK

Spend the next 45 minutes working through the tasks in the "*Session_3_-_MBTA_Exercise*" PDF in the class download folder.

PROJECT WORK

PROJECT WORK

Spend the remaining class time working on developing your mid-term project assessment. This includes importing, understanding, and cleaning your final project data.

Leverage your classmates' intelligence!