Data Wrangling with *dplyr*

INFO 201

Joel Ross

Winter 2017

https://slides.com/joelross/info201w17-dplyr-i/live

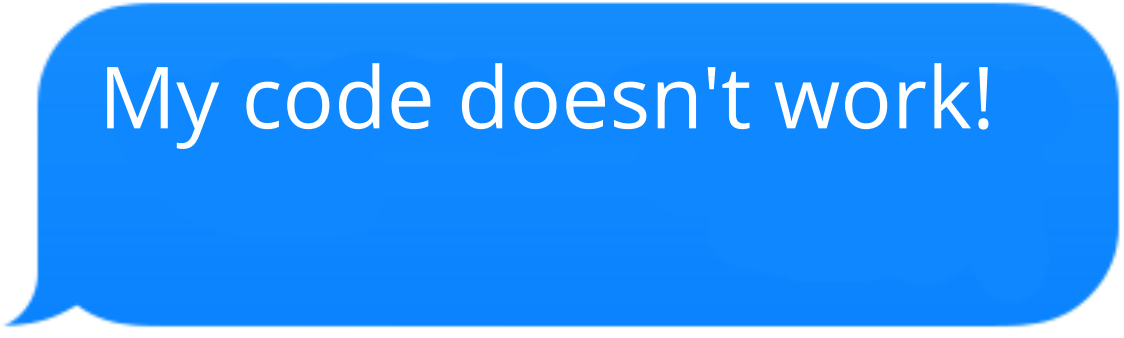
Today's Objectives

By the end of class, you should be able to

- Ask and answer simple questions about data sets
- Use **dplyr**'s *data manipulation grammar* to wrangle data
- Chain methods together using **pipes**

Fixing Bugs

Imagine someone asks you for help:



1. What are you **trying** to achieve?

When I (as a user) do **Foo**, the program should do **Bar**

2. What is **actually** happening?

When I (as a user) do **Foo**, the program does **Baz** instead!

*Need to find "path" from **Foo** to **Bar** ,*
*instead of from **Foo** to **Baz** .*

Problem Solving Activities

1. Reinterpret problem prompt — Understanding!
2. Search for analogous problems
3. Search for solutions — Brainstorming!
4. Evaluate a potential solution
5. Implement a solution — Code Inspection!
6. Evaluate implemented solution — Testing!

Loksa et al. (2016)

What stage are you in?

1. Reinterpret problem prompt
2. Search for analogous problems
3. Search for solutions
4. Evaluate a potential solution
5. Implement a solution
6. Evaluate implemented solution

" *What do you mean by **Foo**?*

" *I want to do **Foo**, but I'm not sure how!*

" *Why might I be getting **Baz** instead of **Bar**?*

Debugging Tips

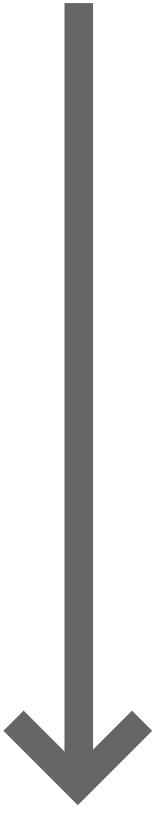
1. Break complicated statements into pieces (with intermediate variables)
    - Especially when filtering or extracting from data frames

2. "Play Computer"

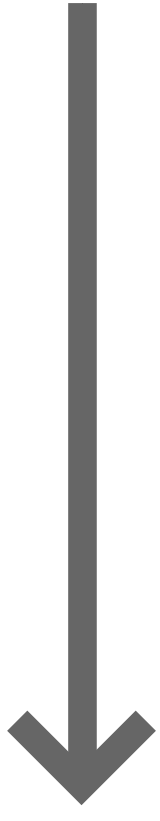
3. **Inspect** (`print()`, `View()`) intermediate variables

4. Compare against known values
    - Sanity check your results (but the answers may surprise you!)

Where to find help



1. Check the notes
   - e.g., the **modules**, slides, your notes, etc
2. Check the documentation
   - e.g., rdocumentation, help functions
3. Check Google/StackOverflow
   - remember to evaluate the answer
4. Ask on Slack! We don't bite!
   - but don't share code please



If stuck (30+ mins no progress), seek help!

Asking Questions of Data Sets

An Example Data Set

	A	B	C	D	E	F
1	cand_nm	contbr_nm	contbr_city	contbr_employer	amount	date
2	Clinton, Hillary Rodham	DISNUTE, CHRISTOPHER	PUYALLUP	N/A	$25	24-Apr-16
3	Sanders, Bernard	KERR, DONNA	SEATTLE	NONE	$27	4-Mar-16
4	Cruz, Rafael Edward 'Ted'	JOHNSON, DAVID	AUBURN	RETIRED	$35	11-Apr-16
5	Sanders, Bernard	LIEBERMAN, DAN	SEATTLE	SMARTTHINGS, INC.	$50	6-Mar-16
6	Clinton, Hillary Rodham	GEORGE, BETTY	KENT	N/A	$55	20-Apr-16
7	Clinton, Hillary Rodham	EULER, JOHN	SEATTLE	HERITAGE BANK	$19	17-Apr-16
8	Sanders, Bernard	LLOYD, LYNN J	LAKEBAY	NOT EMPLOYED	$10	6-Mar-16
9	Clinton, Hillary Rodham	HOLT, JULIE	SHORELINE	SELF-EMPLOYED	$71	20-Apr-16
10	Sanders, Bernard	KOB, L	GIG HARBOR	NOT EMPLOYED	$10	4-Mar-16
11	Cruz, Rafael Edward 'Ted'	KOOY, KYLE MR.	LYNDEN	REICHHARDT & EBE	$25	5-Apr-16
12	Sanders, Bernard	KOB, L	GIG HARBOR	NOT EMPLOYED	$10	6-Mar-16
13	Cruz, Rafael Edward 'Ted'	KOOY, KYLE MR.	LYNDEN	REICHHARDT & EBE	$5	8-Apr-16

*What are three **questions** you could ask about this data set?*

(data source)

Sample Questions

- Who donated the most money?
- Which city did the largest donation come from?
- What was the average donation?

1. Select Columns

Who made the largest donation?

```
donations$contbr_nm
donations$amount
```

What was the average donation for Bernie?

```
$donation$amount
$donations$candidate == "Sanders"
```

Which cities donated in April?

```
donations$contbr_city
grepl("Apr",donations$date) == TRUE
```

2. Filter Rows

Who made the largest donation?

```
donations$contbr_nm[donations$amount == max(donations$amount)]
```

What was the average donation for Bernie?

```
mean(donations$amount[$donations$candidate == "Sanders"])
```

Which cities donated in April?

```
donations$contbr_city[grepl("Apr",donations$date) == TRUE]
```

Grammar for Data Manipulation

Notice the **words** (***verbs***) we used to describe what we did to manipulate the data:

- **Select** the columns of interest
- **Filter** out irrelevant data to keep rows of interest
- **Mutate** a data set by adding more columns
- **Arrange** the rows in a data set
- **Summarize** the data (e.g., calculate the *mean*, *median*, *maximum*, etc).

Module 10 exercise-1

dplyr

dplyr is an R package (library) that implements this Grammar of Data Manipulation.
It provides ***functions*** which mirror the "plain-English" verbs we use to describe data wrangling tasks.

dplyr created by Hadley Wickham

Loading dplyr

dplyr is an external **package**, so needs to be installed and *loaded* into the R environment.

```
# Install `dplyr` package
# Only needs to be done once per machine!
install.packages("dplyr")

# Load the package (tell R functions are available for use)
library("dplyr")
```

https://www.rdocumentation.org/packages/dplyr/versions/0.5.0

dplyr Functions

dplyr functions are named after the grammar of data manipulation. *Note column names have no quotes!*

```
# make the data frame
name <- c('Ada','Bob','Chris','Diya','Emma')
height <- 58:62
weight <- c(115, 117, 120, 123, 126)
my.data <- data.frame(name, height,
                      weight, stringsAsFactors=FALSE)

# use dplyr to get a data frame
# with `name` and `height` cols
select(my.data, name, height)
```

select()

Choose and extract **columns** from a data frame.

storms

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	pressure
Alberto	1007
Alex	1009
Allison	1005
Ana	1013
Arlene	1010
Arthur	1010

```
# Select `storm` and `pressure` cols from `storms` data frame
storm.info <- select(storms, storm, pressure)
```

```
# extract columns by name
storm.info <- storms[, c("storm", "pressure")]  # Note the comma!
```

filter()

Choose and extract **rows** from a data frame.

storms

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	date
Alberto	110	1007	2000-08-12
Allison	65	1005	1995-06-04
Arlene	50	1010	1999-06-13

```
# Select rows with `wind` column greater than or equal to 50
some.storms <- filter(storms, wind >= 50)
```

```
# extract rows by condition
some.storms <- storms[storms$wind >= 50, ]  # Note the comma!
```

mutate()

Add additional **columns** to a data frame. Returns a new frame you should save as a variable!

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	date	ratio
Alberto	110	1007	2000-08-12	9.15
Alex	45	1009	1998-07-30	22.42
Allison	65	1005	1995-06-04	15.46
Ana	40	1013	1997-07-01	25.32
Arlene	50	1010	1999-06-13	20.20
Arthur	45	1010	1996-06-21	22.44

```
# Add `ratio` column that is ratio between pressure and wind
storms <- mutate(storms, ratio = pressure/wind)
```

replace existing data frame

assign new column (like with lists!)

arrange()

Sort the **rows** in the data frame by a *column* value.

storms

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

```
# Arrange storms by INCREASING order of the `wind` column
sorted.storms <- arrange(storms, wind)
```

summarize()

Generate a data frame that has an aggregation of a particular **column**.

city	particle size	amount ($\mu g/m^3$)
New York	large	23
New York	small	14
London	large	22
London	small	16
Beijing	large	121
Beijing	small	56

median
22.5

```
# Compute the median value of the `amount` column
summary <- summarize(pollution, median = median(amount))
```

Module 10 exercise-2

Multiple Operations

Asking questions often involves multiple steps, meaning we need to take the results of one **verb** (function) and pass it in to the next.

For example, there are a few steps to ask the following question about the `mtcars` data set:

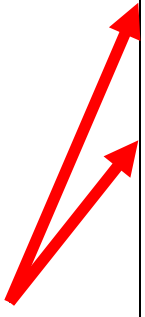
> *Which 4-cylinder car gets the best milage per gallon?*

1. **filter** down the data set to only 4-cylinder cars
2. *of those cars*, **filter** down to the one with the highest mpg
3. *of that car*, **select** the car name

Temporary Variables

So far we've been using ***temporary, intermediate variables*** to perform these multiple steps.

```
# Preparation: load the data set
data("mtcars")

# Preparation: add a column that is the car name
mtcars.named <- mutate(mtcars, car.name = row.names(mtcars))


# 1. Filter down to only four cylinder cars
four.cyl <- filter(mtcars.named, cyl == 4)

# 2. Filter down to the one with the highest mpg
best.four.cyl <- filter(four.cyl, mpg == max(mpg))

# 3. Select the car name of the car
best.car.name <- select(best.four.cyl, car.name)
```

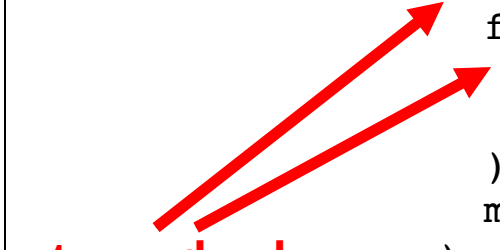


temp variable

Function Nesting

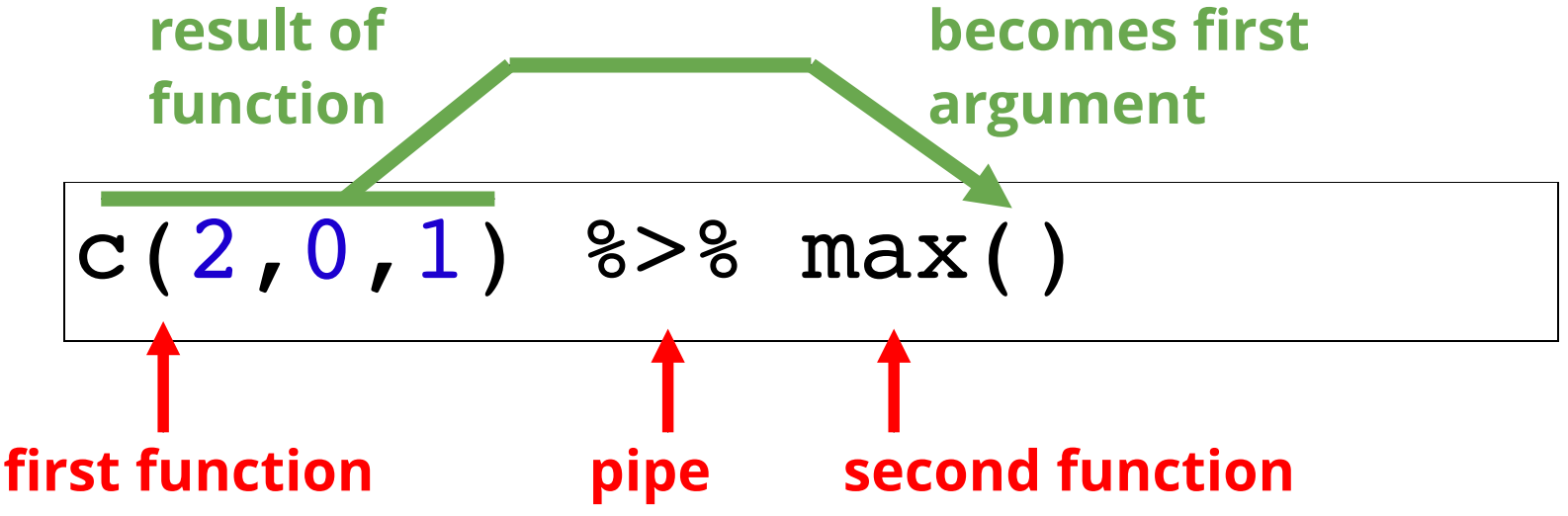
We could also **nest** the function calls, using the returned values as **anonymous variables**.

```
# Preparation: load the data set
data("mtcars")

# Preparation: add a column that is the car name
mtcars.named <- mutate(mtcars, car.name = row.names(mtcars))


# Write a nested operation to return the best car name
best.car.name <- select(  # 3. Select car name of the car
                    filter(  # 2. Filter down to one with the highest mpg
                      filter( # 1. Filter down to only four cylinder cars
                        mtcars.named,  # parameters for the Step 1 filter
                        cyl == 4
                      ),
                      mpg == max(mpg)  # other args for the Step 2 filter
                    ),
                    car.name  # other args for the Step 3 select
                  )
```



Pipe Operator

dplyr provides a **pipe operator** (`%>%`) that takes the result of the first function and sets it as the first argument of the second function.



Pipe Operator

dplyr provides a **pipe operator** (`%>%`) that takes the result of the first function and sets it as the first argument of the second function.

```
# Preparation: load the data set
data("mtcars")

# Preparation: add a column that is the car name
mtcars.named <- mutate(mtcars, car.name = row.names(mtcars))


# use pipes to "chain" functions
best.car.name <- filter(mtcars.named, cyl == 4) %>%  # Step 1
  filter(mpg == max(mpg)) %>%  # Step 2
  select(car.name)  # Step 3
```

Module 10 exercise-3

Action Items!

- Be comfortable with **modules 9**
- Assignment 4 due ***Tuesday before class***

Thursday: **dplyr** and multiple data frames!