Data Transformation: Slicing Your Data

Patrick Mathias

December 13, 2020

Presentation adapted from…

Amrom Obstfeld

Assistant Professor of Clinical Pathology and Laboratory Medicine

University of Pennsylvania Perelman School of Medicine

Director of Hematology and Coagulation Laboratories

Children's Hospital of Philadelphia

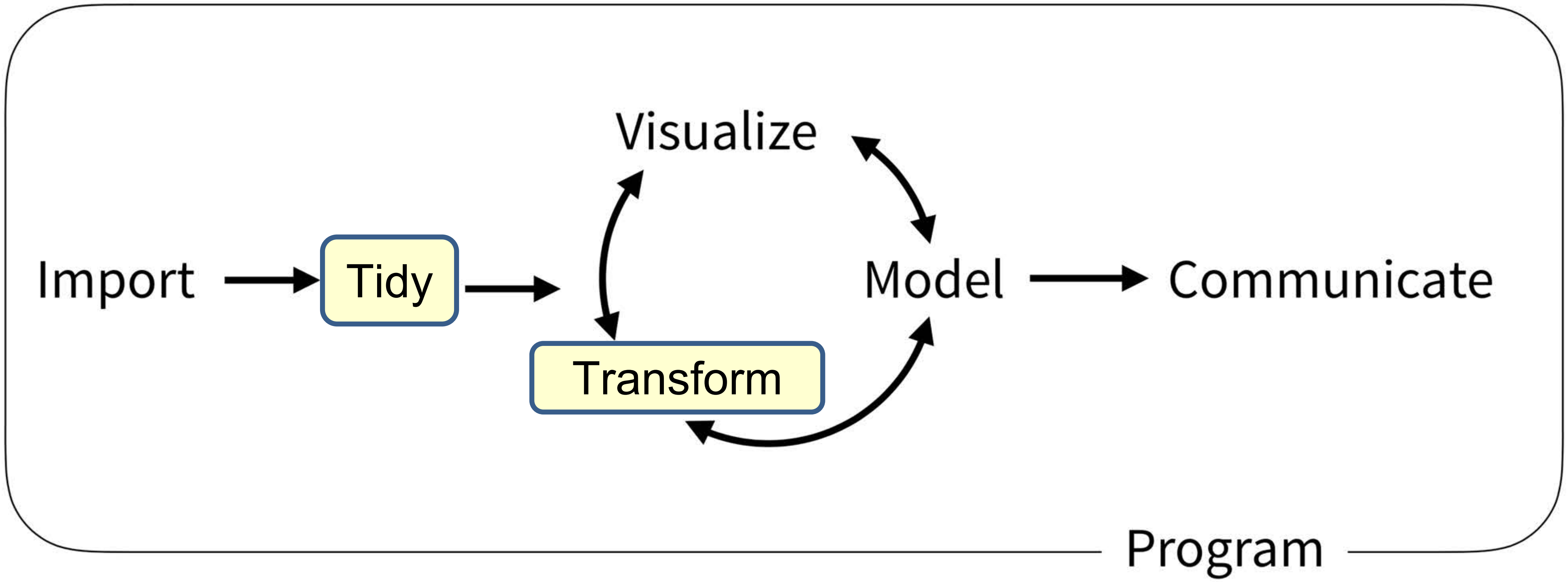
Goal

1. Learn how to use dplyr to transform data frames

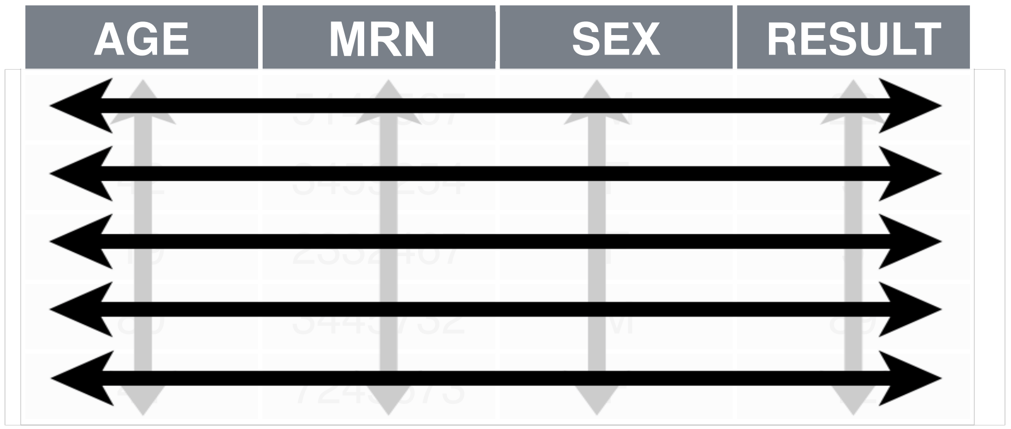
Objectives

1. List the major forms of data transformation implemented in dplyr
2. Use code templates with dplyr functions to tidy a raw data set

Typical Data Science Pipeline



What is a "Tidy" Data Frame



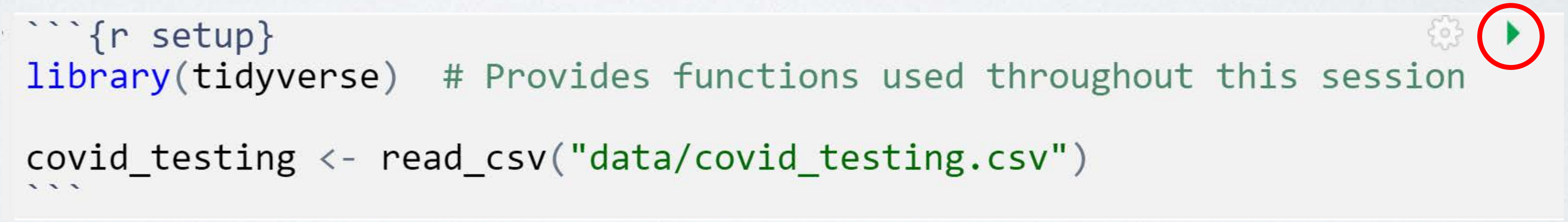
A data set is **tidy** if:

1. Each **variable** is in its own **column**
2. Each **observation** is in its own **row**
3. Each **value** is in its own **cell**

Your Turn 1

Open **"04-Transform.Rmd"**

Run the setup chunk



~~~
```{r setup}
library(tidyverse)  # Provides functions used throughout this session

covid_testing <- read_csv("data/covid_testing.csv")
```
~~~

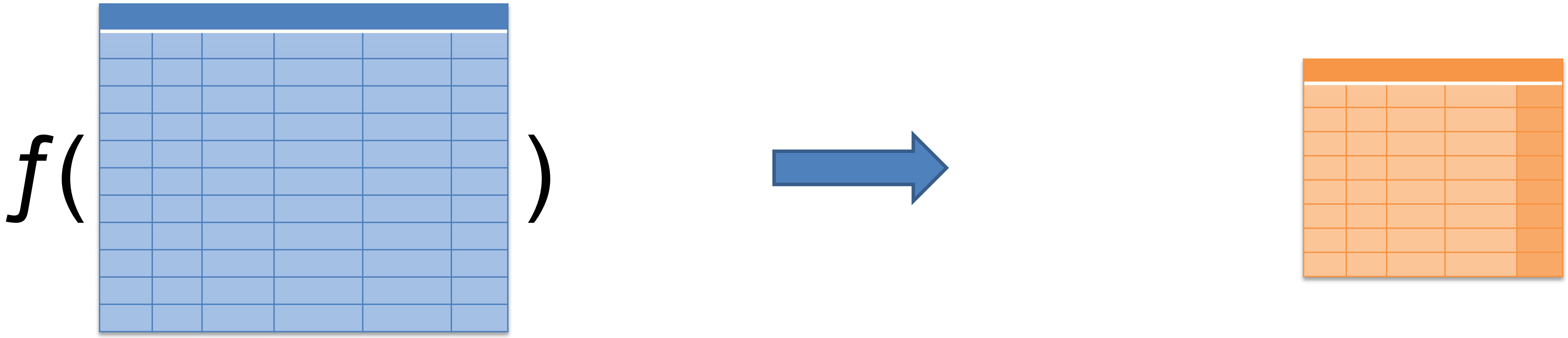
Transform Data with

dplyr

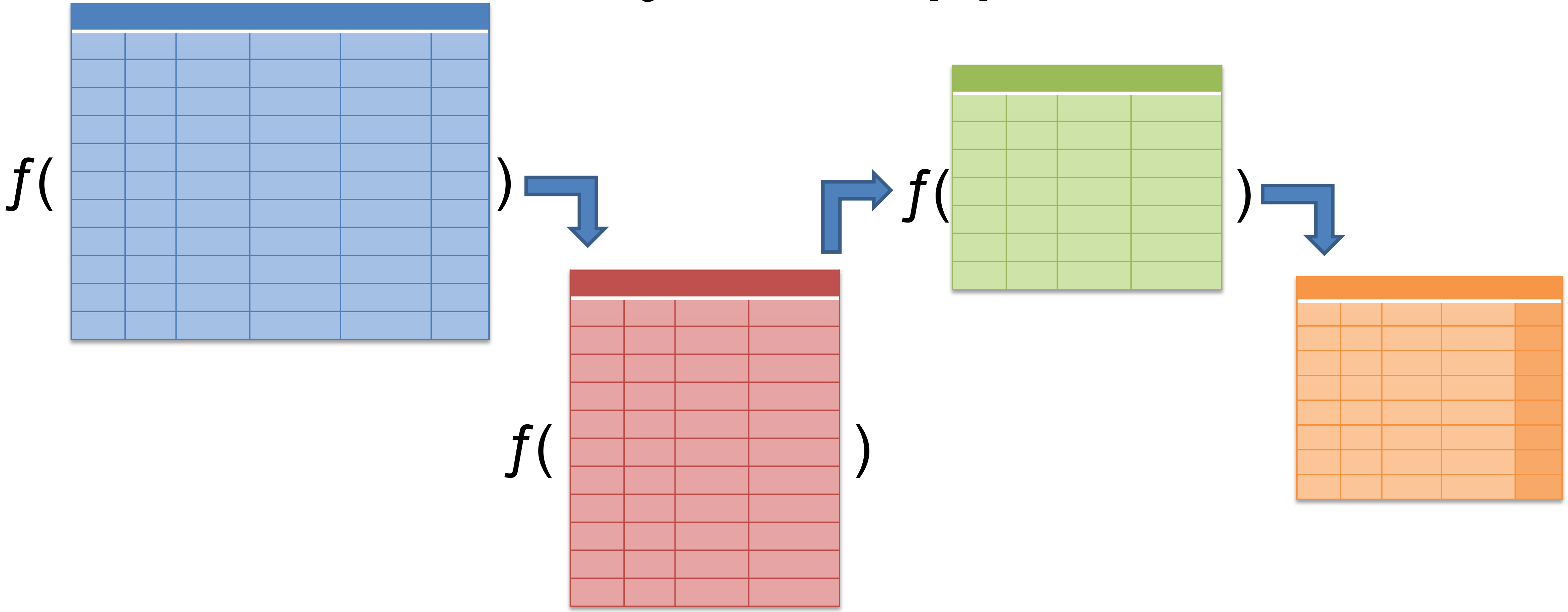
dplyr

dplyr implements a *grammar* for transforming tabular data.

Analytical Approach



Analytical Approach

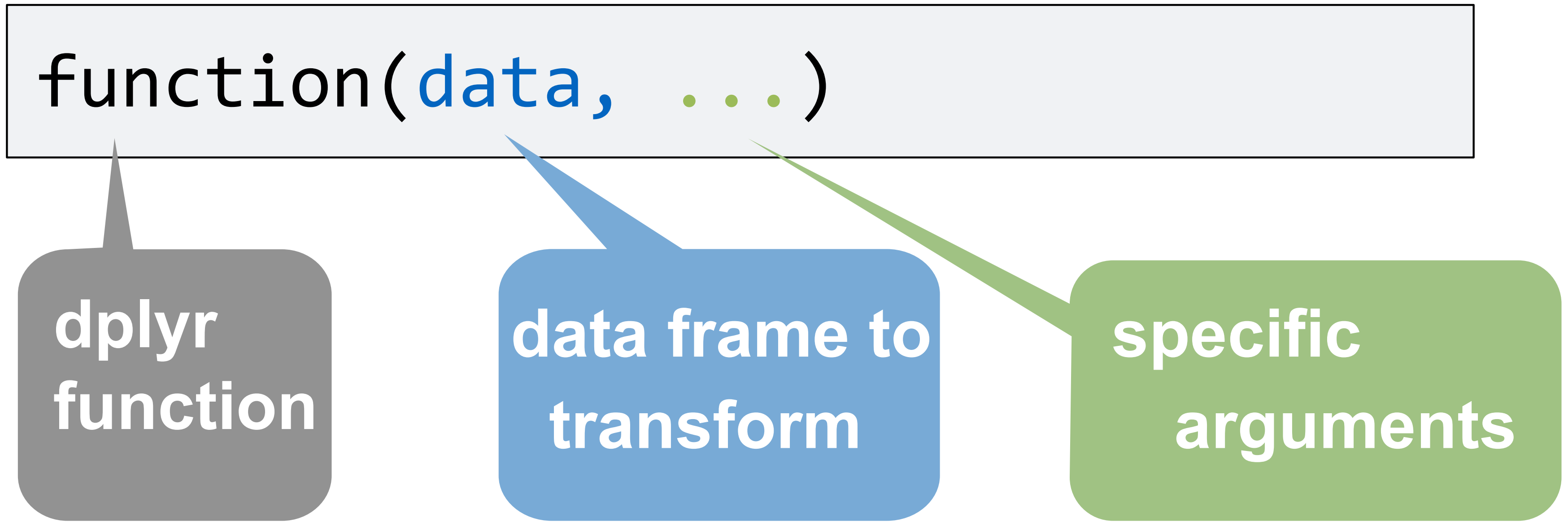


dplyr: a grammar for transforming data

1	**Choose** columns.	select()
2	**Extract** rows.	filter()
3	**Derive** new columns.	mutate()
4	**Change** the unit of analysis.	summarize()

Common syntax

Each function takes a data frame as its first argument and returns a data frame as its output.



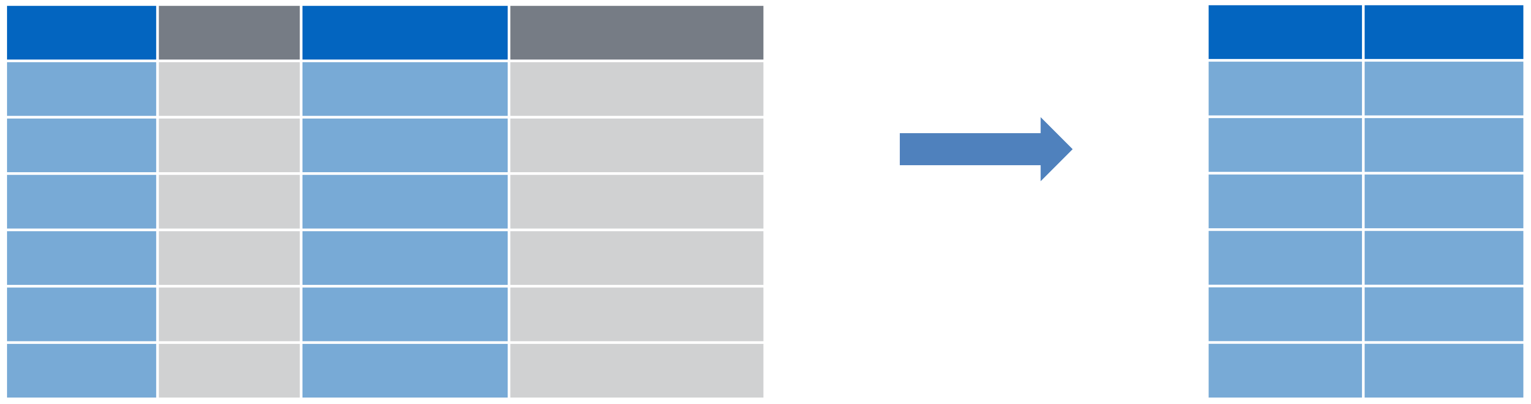
```
function(data, ...)
```

Pulling specific columns out of your data frame

select()

select()

Extract columns from a data frame

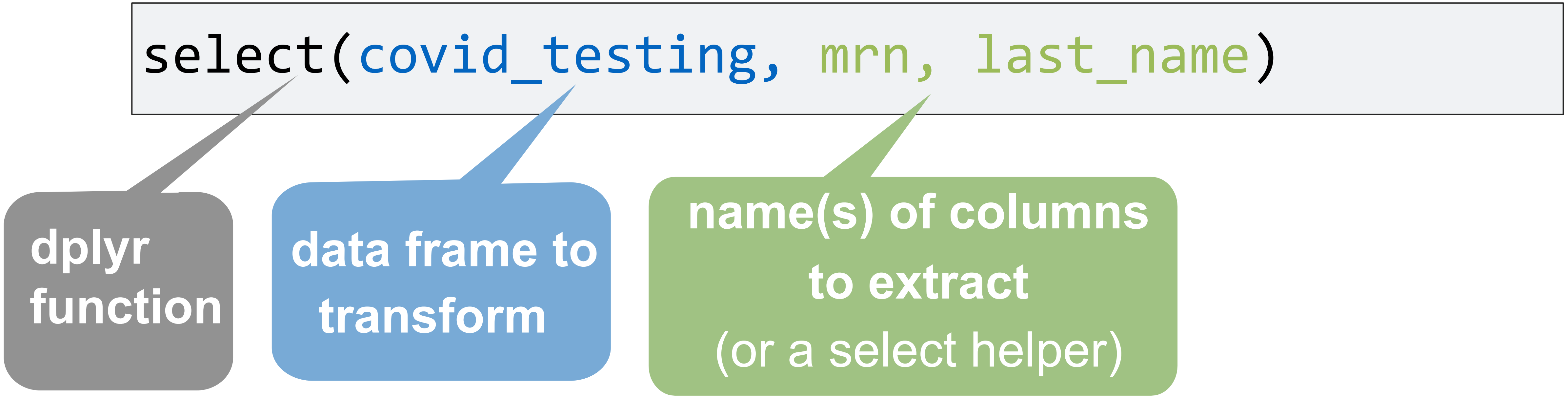


= Number of rows

↓ Number of Columns

select()

Extract columns from a data frame



```r
select(covid_testing, mrn, last_name)
```

select()

Extract columns from a data frame **by name**

```
select(covid_testing, mrn, last_name)
```

covid_testing

mrn	first_name	last_name	gender
5000876	sarella	stark	female
5006017	alester	stark	male
5001412	jhezane	westerling	female
5000533	penny	targaryen	female

...

→

mrn	last_name
5000876	stark
5006017	stark
5001412	westerling
5000533	targaryen

select()

Extract columns from a data frame **by name**

```
select(covid_testing, -mrn, -last_name)
```

covid_testing

mrn	first_name	last_name	gender
5000876	sarella	stark	female
5006017	alester	stark	male
5001412	jhezane	westerling	female
5000533	penny	targaryen	female

...

→

first_name	gender
sarella	female
alester	male
jhezane	female
penny	female

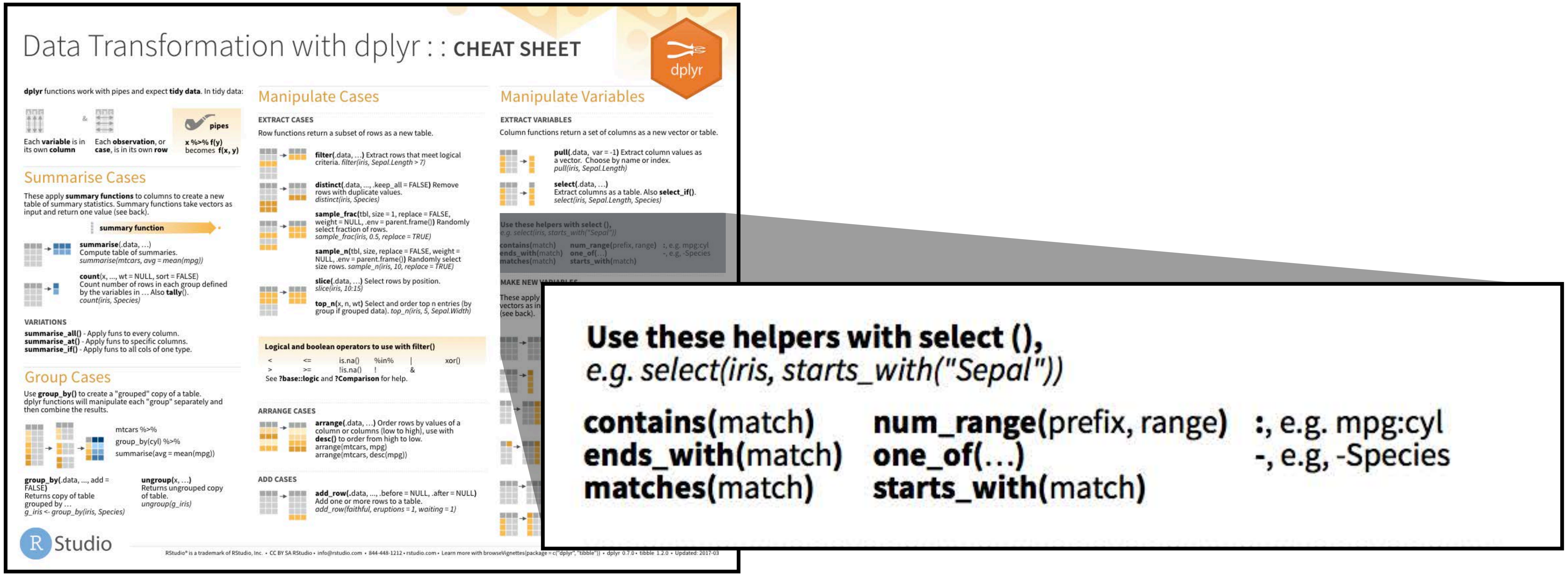
Your Turn 2

- Alter the code to select just the `first_name` column from `covid_testing`

- If you have time, try to remove the `first_name` column

```
covid_testing_2 <- select(covid_testing, _____)
```

select() helpers



Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column** & Each **observation**, or **case**, is in its own **row**

pipes
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

summarise(.data, …)
Compute table of summaries.
summarise(mtcars, avg = mean(mpg))

count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

```
mtcars %>%
group_by(cyl) %>%
summarise(avg = mean(mpg))
```

group_by(.data, ..., add = FALSE**)**
Returns copy of table grouped by …
g_iris <- group_by(iris, Species)

ungroup(x, …**)**
Returns ungrouped copy of table.
ungroup(g_iris)

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …**)** Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE**)** Remove rows with duplicate values.
distinct(iris, Species)

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …**)** Select rows by position.
slice(iris, 10:15)

top_n(x, n, wt**)** Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …**)** Order rows by values of a column or columns (low to high), use with **desc()** to order from high to low.
arrange(mtcars, mpg)
arrange(mtcars, desc(mpg))

ADD CASES

add_row(.data, ..., .before = NULL, .after = NULL**)**
Add one or more rows to a table.
add_row(faithful, eruptions = 1, waiting = 1)

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

pull(.data, var = -1**)** Extract column values as a vector. Choose by name or index.
pull(iris, Sepal.Length)

select(.data, …**)**
Extract columns as a table. Also **select_if()**.
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

contains(match)	**num_range**(prefix, range)	:, e.g. mpg:cyl
ends_with(match)	**one_of**(…)	-, e.g, -Species
matches(match)	**starts_with**(match)	

MAKE NEW

These apply
vectors as in
(see back).

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

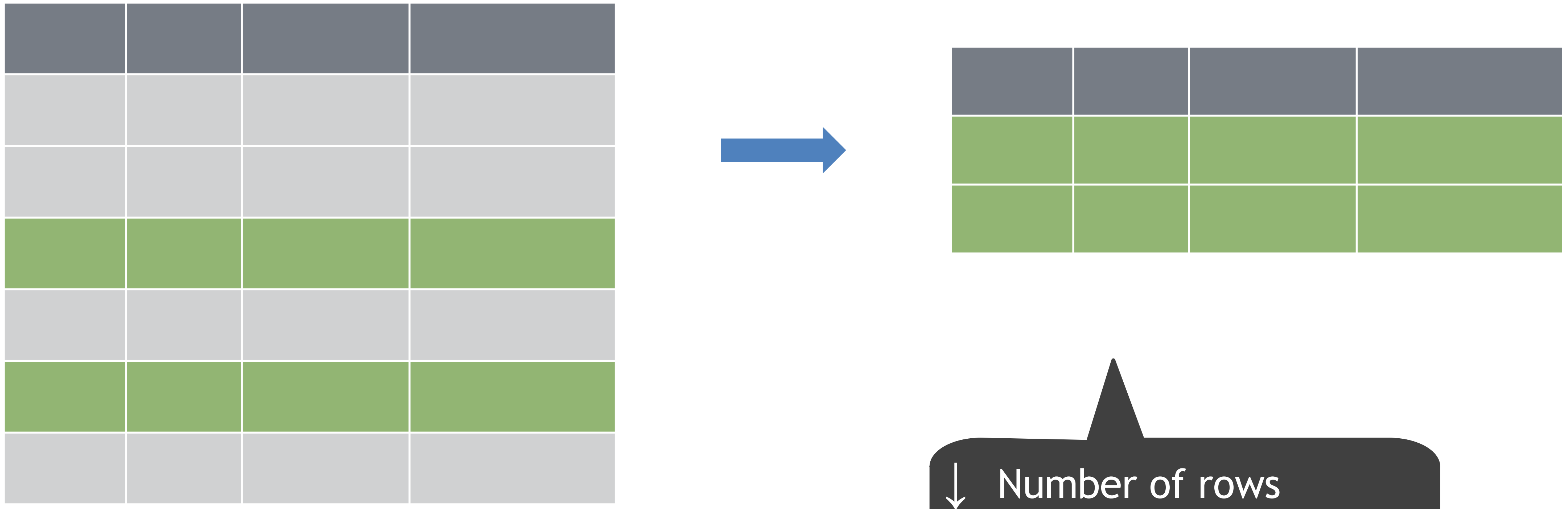
contains(match)	**num_range**(prefix, range)	:, e.g. mpg:cyl
ends_with(match)	**one_of**(…)	-, e.g, -Species
matches(match)	**starts_with**(match)	

Pulling specific rows out of your data frame

filter()

filter()

Extract rows that meet logical criteria

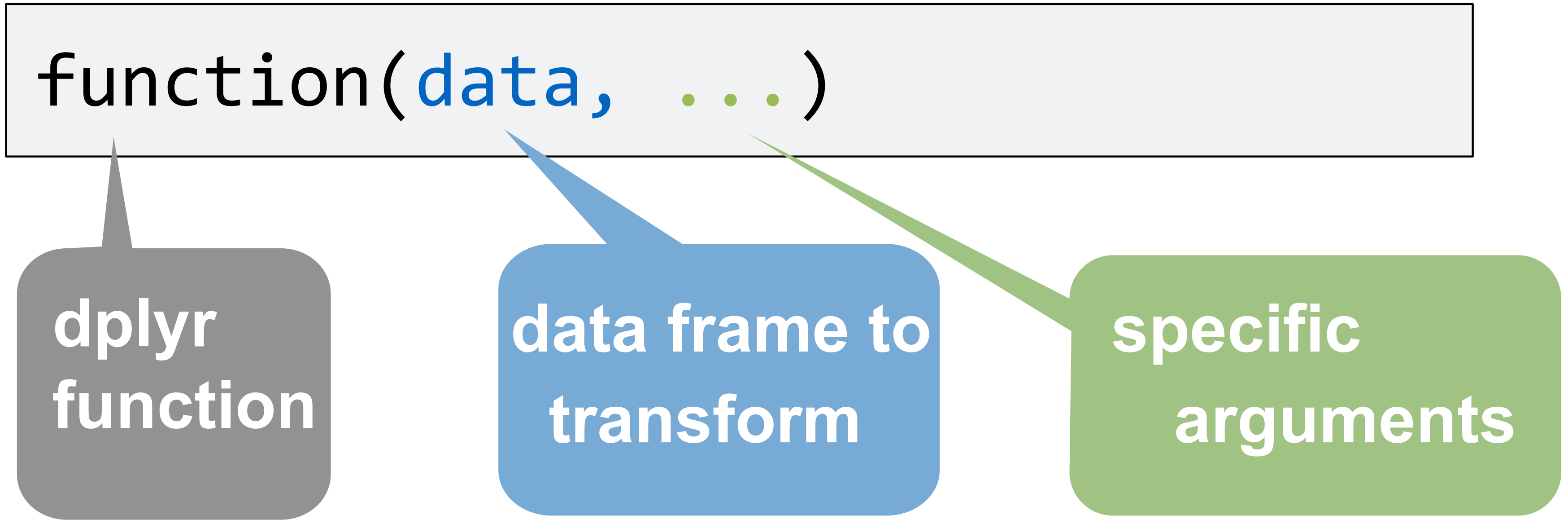


↓ Number of rows

= Number of Columns

Common syntax

Each function takes a data frame as its first argument and returns a data frame as its output.



```
function(data, ...)
```

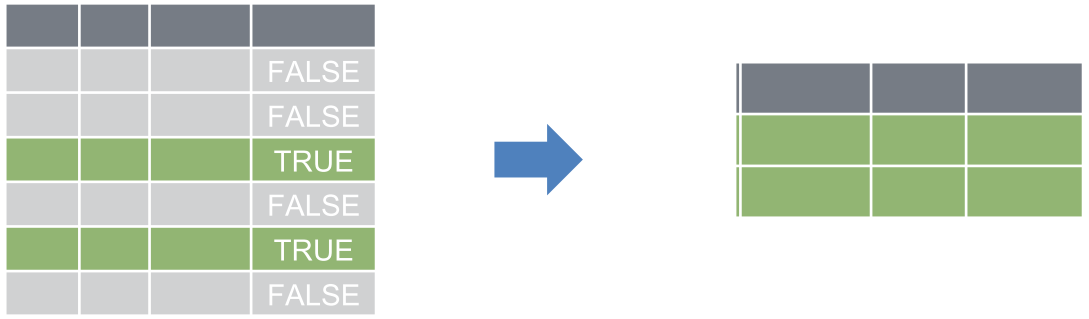
filter()

Extract rows that meet logical criteria

```
filter(data,... )
```

data frame to transform

one or more logical tests
(filter returns each row for which the test is TRUE)

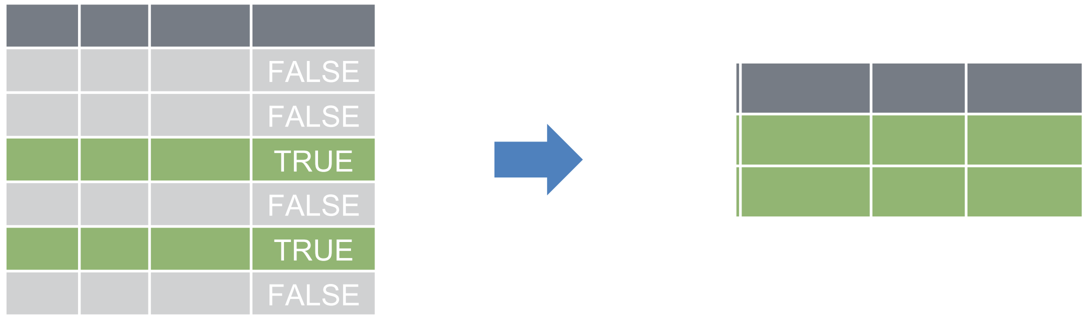


filter()

Extract rows that meet logical criteria

```
filter(data, column_name == criteria )
```

one or more logical tests
(filter returns each row for which the test is TRUE)



filter()

Extract rows that meet logical criteria

```
filter(covid_testing, mrn==5000083)
```

	mrn	first_name	last_name
FALSE	5000876	sarella	stark
FALSE	5006017	alester	stark
FALSE	5001412	jhezane	westerling
TRUE	5000083	lollys	clegane

→

mrn	first_name	last_name
5000083	lollys	clegane

filter()

Extract rows that meet logical criteria

```
filter(covid_testing, mrn==5000083)
```

= sets

(returns nothing)

== tests if equal

(returns TRUE or FALSE)

mrn	first_name	last_name
5000876	sarella	stark
5006017	alester	stark
5001412	jhezane	westerling
5000083	lollys	clegane

filter()

Values coded as character strings must be surrounded by quotes

Extract rows that meet logical criteria.

```
filter(covid_testing, last_name=="stark")
```

mrn	first_name	last_name	
5000876	sarella	stark	TRUE
5006017	alester	stark	TRUE
5001412	jhezane	westerling	FALSE
5000083	lollys	clegane	FALSE

→

mrn	first_name	last_name
5000876	sarella	stark
5006017	alester	stark

dplyr

filter()

Extract rows that meet logical criteria

```
filter(data, ...              )
```

data frame to transform

one or more logical tests
(filter returns each row for which the test is TRUE)

Logical tests

`x < y`	Less than
`x > y`	Greater than
`x == y`	Equal to
`x <= y`	Less than or equal to
`x >= y`	Greater than or equal to
`x != y`	Not equal to
`x %in% y`	Group membership
`is.na(x)`	Is NA
`!is.na(x)`	Is not NA

Pop Quiz

What is the result?

```
1 == 1
```

Pop Quiz

What is the result?

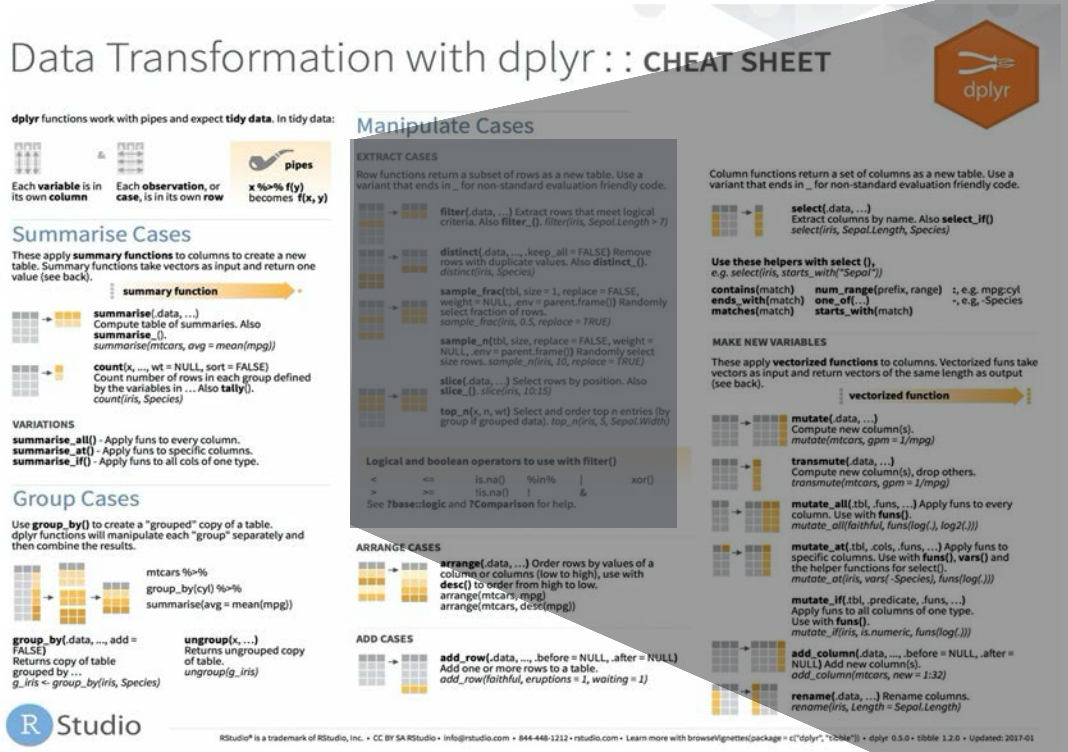
```
3 != 1
```

Your Turn 3

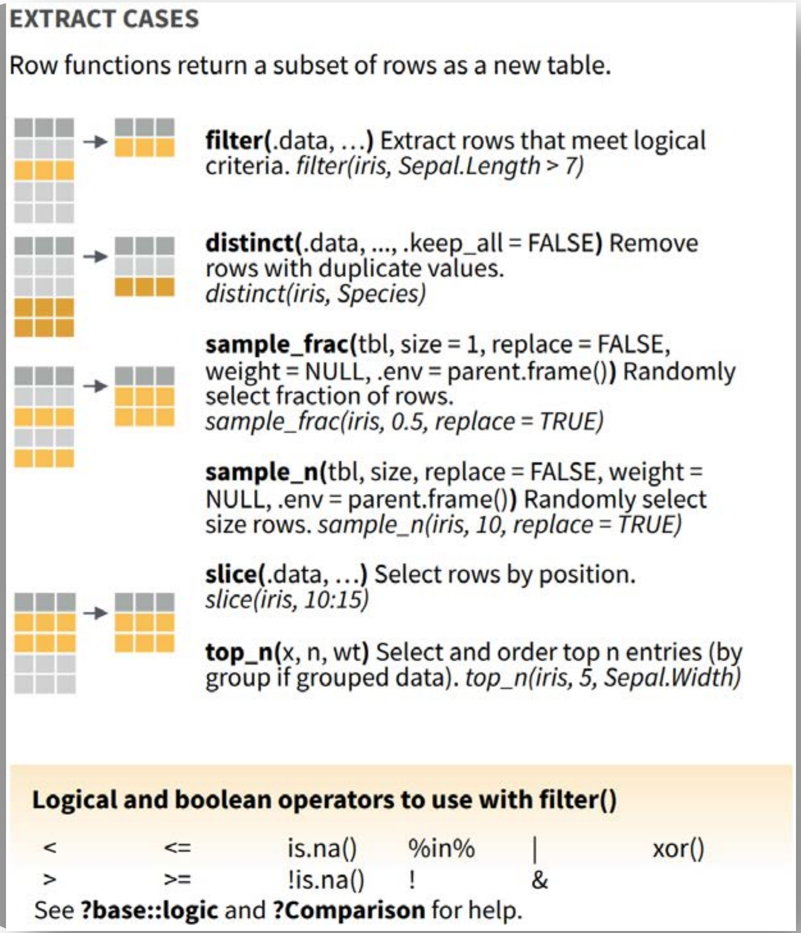
Use filter() with the logical operators to find:

- Every test for patients **over age 80**
- All of the covid testing where the demographic group (demo_group) is **equal to "client"**
- CHALLENGE:
  - All of the covid testing where the patient class (patient_class) **is NA** [Hint: See slide titled "Logical Tests"]

filter() variants



Data Transformation with dplyr : : CHEAT SHEET



EXTRACT CASES

Row functions return a subset of rows as a new table.

filter(.data, …**)** Extract rows that meet logical criteria. *filter(iris, Sepal.Length > 7)*

distinct(.data, ..., .keep_all = FALSE**)** Remove rows with duplicate values. *distinct(iris, Species)*

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select fraction of rows. *sample_frac(iris, 0.5, replace = TRUE)*

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*

slice(.data, …**)** Select rows by position. *slice(iris, 10:15)*

top_n(x, n, wt**)** Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

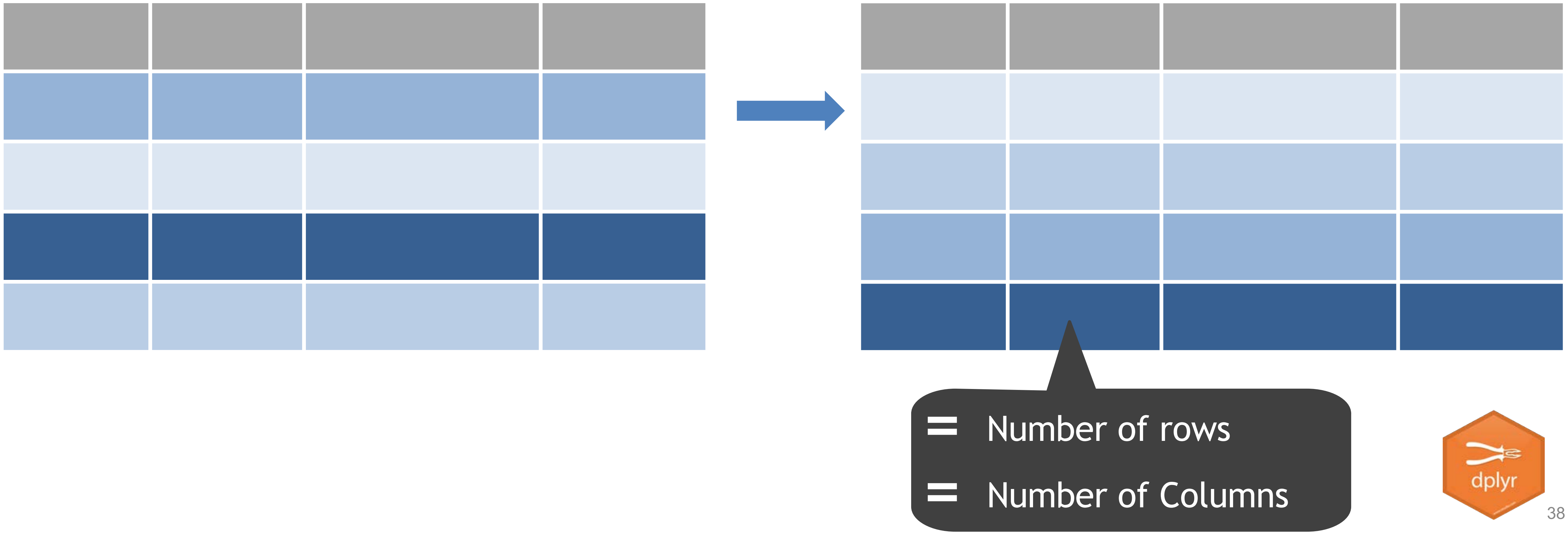
See **?base::logic** and **?Comparison** for help.

What else?

arrange()

arrange()

Order rows by values in a column



arrange()

Order rows by values in a column

```
arrange(data,... )
```

data frame to transform

name(s) of columns to arrange by

arrange()

Order rows by values in a column

```
arrange(covid_testing, first_name)
```

mrn	first_name	last_name
5000876	sarella	stark
5006017	alester	stark
5001412	jhezane	targaryen
5000533	penny	targaryen

→

mrn	first_name	last_name
5006017	alester	stark
5001412	jhezane	targaryen
5000533	penny	targaryen
5000876	sarella	stark

arrange()

Order rows by values in a column

```
arrange(covid_testing, desc(mrn))
```

mrn	first_name	last_name
5000876	sarella	stark
5006017	alester	stark
5001412	jhezane	targaryen
5000533	penny	targaryen

→

mrn	first_name	last_name
5006017	alester	stark
5001412	jhezane	targaryen
5000876	sarella	stark
5000533	penny	targaryen

Your Turn 4

The column `ct_result` contains the cycle threshold (Ct) for the real-time PCR that generated the final result.

How might you use `arrange()` to determine the highest and lowest Ct result in the dataset?

Pop Quiz

The default behavior of arrange() is to order from lower to higher values.

When might arrange() place "1000" before "50"?

Goal

1. Learn how to use dplyr to transform data frames

Objectives

1. List the major forms of data transformation implemented in dplyr
2. Use code templates with dplyr functions to tidy a raw data set