

# Statistical Analyses in R

Session 5  
Dan Herman  
July 17, 2020

July 16 2020	<b>Session</b>	<b>Instructor</b>
1:00 pm - 1:30 pm	Instructor Introductions, Introduction to technology	Amrom Obstfeld
1:30 pm - 2:15 pm	Introduction to R and RStudio	Joe Rudolf
2:30 pm - 3:15 pm	R basics for Reproducible Reporting	Patrick Mathias
3:30 pm - 5:00 pm	Data Visualization in R	Stephan Kadauke
July 17 2020		
1:00 pm - 2:30 pm	Data Transformation	Amrom Obstfeld
2:45 pm - 4:15 pm	Statistical Analysis in R	Dan Herman
4:30 pm - 5:00 pm	Advanced Reporting in R	Patrick Mathias

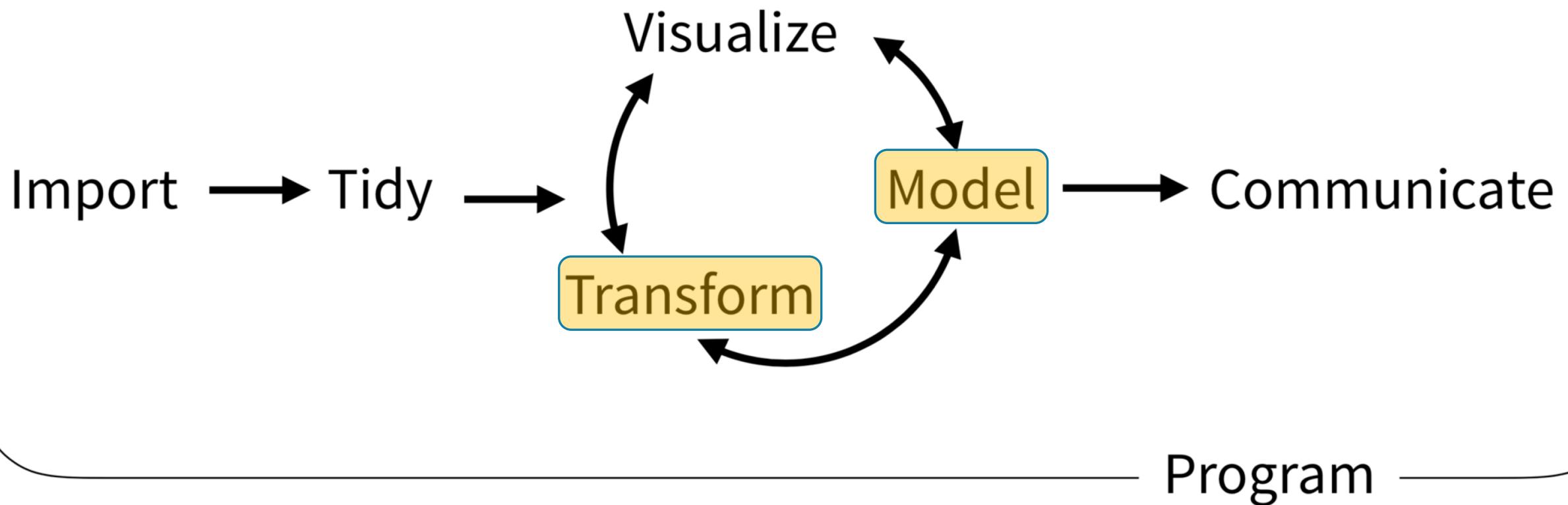
# Goals

1. Learn how to summarize and assess data

# Objectives

1. Calculate a summary statistic for a variable using `summarize`
2. Calculate of summary statistic for a variable separately for a group of observations, using `group\_by` and `summarize`
3. Perform a simple test for association

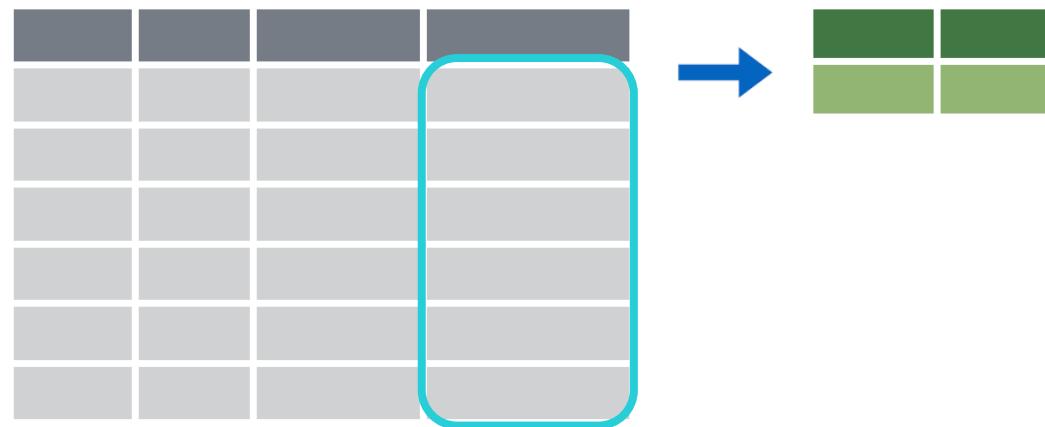
# Typical Data Science Pipeline



# Summarize()

# summarize()

- Make summaries of your data



# summarize()

- Make summaries of your data

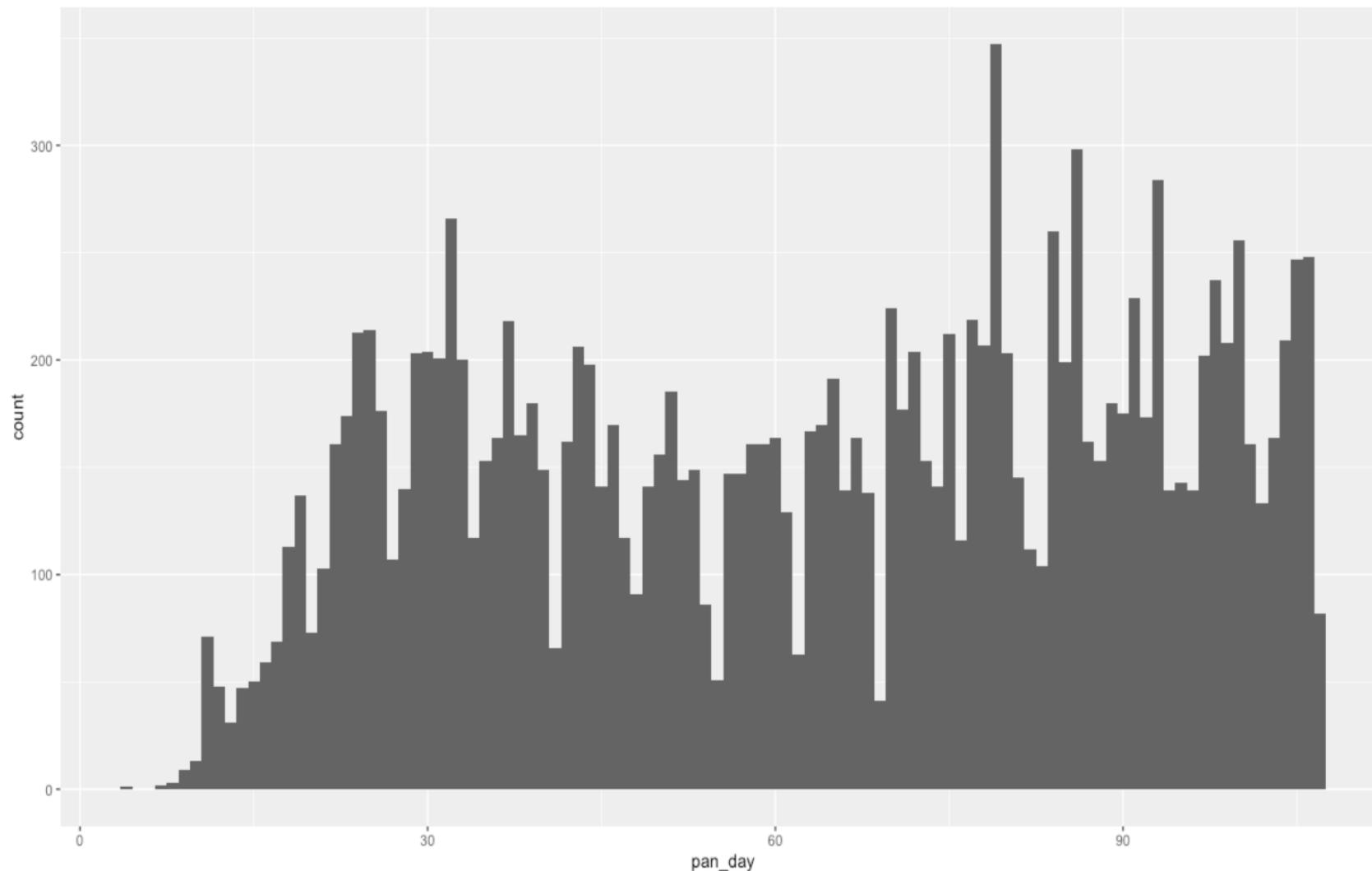
```
covid_testing %>%  
  summarize(new_variable = calculation)
```

name for new  
variable

Value or  
function



# Q: How many tests are ordered per day?



# summarize()

- Make summaries of your data

```
covid_testing %>%  
  select(mrn, pan_day) %>%  
  head(4) %>%  
  
  summarize(order_count = n())
```

function that returns  
number of observations

mrn	pan_day
5001412	4
5000533	7
5009134	7
5008518	8



order_count
4



# summarize()

- Make summaries of your data

```
covid_testing %>%  
  select(mrn, pan_day) %>%  
  head(4) %>%  
  
  summarize(order_count = n(),  
            day_count = n_distinct(pan_day))
```

function that returns  
number of distinct values

mrn	pan_day
5001412	4
5000533	7
5009134	7
5008518	8



order_count	day_count
4	3

# Your Turn 1

- Open “05– Stats.Rmd”
- Run the setup chunk
- Fill-in gaps to calculate:
  - a) Mean count of orders per `pan\_day`
  - b) Mean count of orders per clinic



## Vector Functions

### TO USE WITH MUTATE()

#### COUNTS

`dplyr::n()` - number of values/rows  
`dplyr::n_distinct()` - # of uniques  
`sum(!is.na())` - # of non-NA's

#### LOCATION

`mean()` - mean, also `mean(!is.na())`  
`median()` - median

#### LOGICALS

`mean()` - Proportion of TRUE's  
`sum()` - # of TRUE's

#### POSITION/ORDER

`dplyr::first()` - first value  
`dplyr::last()` - last value  
`dplyr::nth()` - value in nth location of vector

#### RANK

`quantile()` - nth quantile  
`min()` - minimum value  
`max()` - maximum value

#### SPREAD

`IQR()` - Inter-Quartile Range  
`mad()` - median absolute deviation  
`sd()` - standard deviation  
`var()` - variance

## Summary Functions

### TO USE WITH SUMMARISE()

`summarise()` applies summary functions to columns to create a new table. Summary functions take vectors as input and return single values as output.

#### summary function

##### COUNTS

`dplyr::n()` - number of values/rows  
`dplyr::n_distinct()` - # of uniques  
`sum(!is.na())` - # of non-NA's

##### LOCATION

`mean()` - mean, also `mean(!is.na())`  
`median()` - median

##### LOGICALS

`mean()` - Proportion of TRUE's  
`sum()` - # of TRUE's

##### POSITION/ORDER

`dplyr::first()` - first value  
`dplyr::last()` - last value  
`dplyr::nth()` - value in nth location of vector

##### RANK

`quantile()` - nth quantile  
`min()` - minimum value  
`max()` - maximum value

##### SPREAD

`IQR()` - Inter-Quartile Range  
`mad()` - median absolute deviation  
`sd()` - standard deviation  
`var()` - variance

## Row Names

Tidy data does not use rownames, which store a variable outside of the columns. To work with the rownames, first move them into a column.

`rownames_to_column()`  
Move row names into col.  
`a <- rownames_to_column(iris, var = "C")`

`column_to_rownames()`  
Move col in row names.  
`column_to_rownames(a, var = "C")`

Also `has_rownames()`, `remove_rownames()`

## Combine Tables

### COMBINE VARIABLES

x	y
A B C	A B B
a t 1	a t 3
b u 2	b u 2
c v 3	d w 1

Use `bind_cols()` to paste tables beside each other as they are.

`bind_cols(...)` Returns tables placed side by side as a single table.  
BE SURE THAT ROWS ALIGN.

Use a "Mutating Join" to join one table to columns from another, matching values with the rows that they correspond to. Each join retains a different combination of values from the tables.

x	y	by = NULL,
A B C D	a t 1 3	copy=FALSE, suffix=c("x","y"),...)
a t 1 3	b u 2 2	Join matching values from y to x.

x	y	by = NULL,
A B C D	a t 1 3	copy = FALSE, suffix=c("x","y"),...)
b u 2 2	d w N A 1	Join matching values from x to y.

x	y	by = NULL,
A B C D	a t 1 3	copy = FALSE, suffix=c("x","y"),...)
b u 2 2	c v 3 N A	Join data. Retain only rows with matches.

x	y	by = NULL,
A B C D	a t 1 3	copy=FALSE, suffix=c("x","y"),...)
b u 2 2	d w N A 1	Join data. Retain all values, all rows.

x	y	by = c("col1", "col2")
A B C D	a t 1 3	to specify the column(s) to match on. <code>left_join(x, y, by = "A")</code>

x	y	by = c("col1" = "col2")
A B C D	a t 1 d w	to match on columns with different names in each data set. <code>left_join(x, y, by = "C" = "D")</code>

x	y	by = c("C" = "D", suffix = "1", "2")
A B C D	a t 1 d w	Use <code>suffix</code> to specify suffix to give to duplicate column names. <code>left_join(x, y, by = c("C" = "D", suffix = "1", "2"))</code>

### COMBINE CASES

x	y
A B C	a t 1
b u 2	c v 3

Use `bind_rows()` to paste tables below each other as they are.

x	y
A B C	a t 1
b u 2	c v 3
d w 4	

x	y
A B C	a t 1
b u 2	

x	y
A B C	a t 1
b u 2	b u 2
c v 3	

Use `setequal()` to test whether two data sets contain the exact same rows (in any order).

### EXTRACT ROWS

x	y
A B C	a t 1
b u 2	c v 3

Use a "Filtering Join" to filter one table against the rows of another.

x	y
A B C	a t 1

x	y
A B C	a t 1
b u 2	b u 2
c v 3	

# summarize() examples

- Last pandemic day (in data)
- Median turnaround time



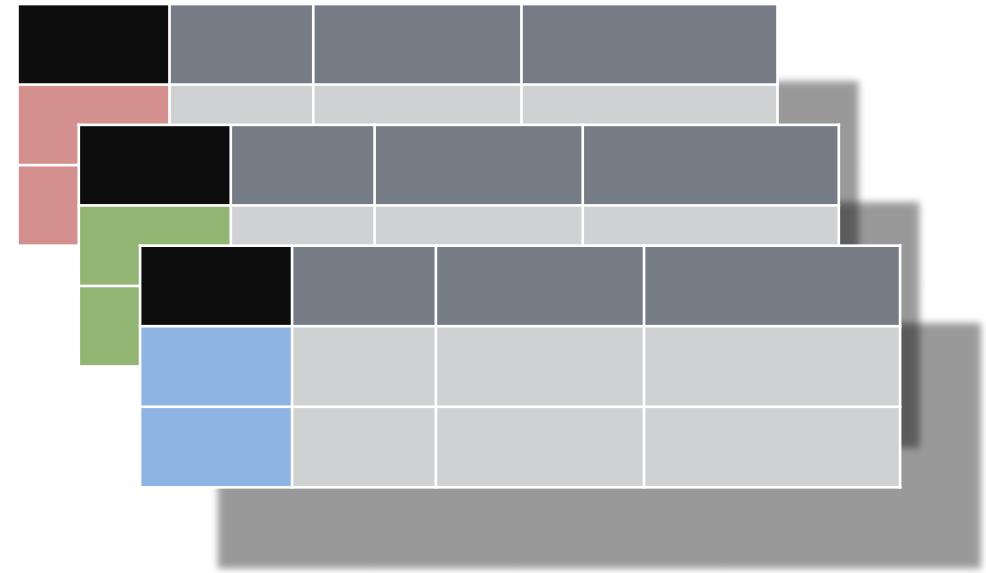
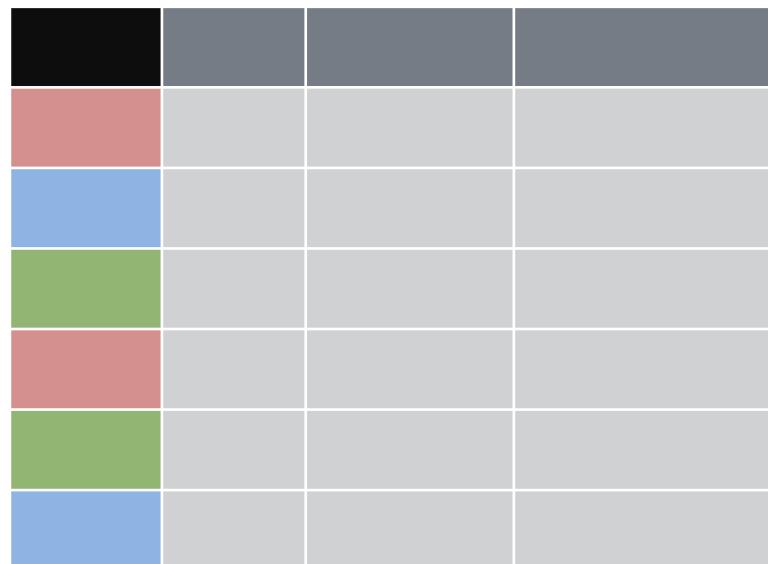
# Your Turn 2

Consider:

How would you calculate the median number of orders per day?

**group\_by()**

# group\_by()



# group\_by()

- *Grouping observations based on a specific variable's values*

```
covid_testing %>%  
  group_by(variable)
```



name of variable  
to group by



# group\_by()

- *Group observations by pan\_day*

```
covid_testing %>%  
  group_by(pan_day)
```

```
# A tibble: 15,524 x 17  
# Groups:   pan_day [102]  
  mrn first_name last_name gender pan_day  
  <dbl> <chr>     <chr>    <chr>   <dbl>  
1 5.00e6 jhezane   westerli... female    4  
2 5.00e6 penny     targaryen female    7  
3 5.01e6 grunt     rivers    male      7  
4 5.01e6 melisandre swyft    female    8  
5 5.01e6 rolley    karstark  male      8
```



# group\_by()

- *Group observations by `pan\_day` and `clinic\_name`*

```
covid_testing %>%  
  select(mrn, pan_day, clinic_name) %>%  
  group_by(pan_day, clinic_name)
```

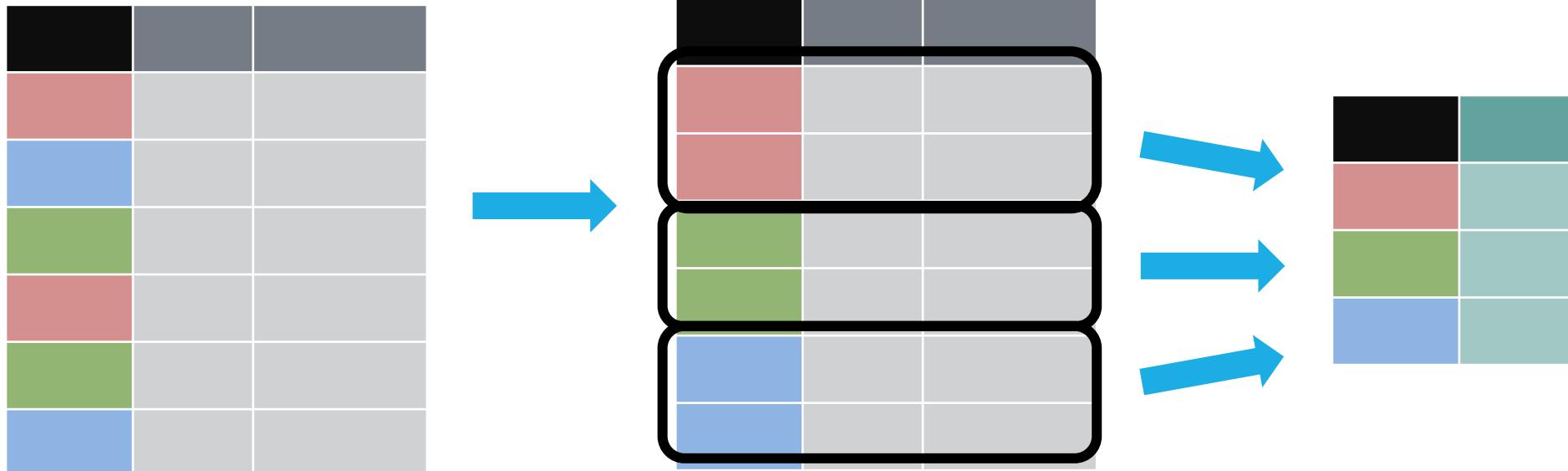
```
# A tibble: 15,524 x 3  
# Groups:   pan_day, clinic_name [2,526]  
  mrn    pan_day clinic_name  
  <dbl>    <dbl> <chr>  
1 5001412      4 inpatient ward a  
2 5000533      7 clinical lab  
3 5009134      7 clinical lab  
4 5008518      8 clinical lab  
5 5008967      8 emergency dept
```



```
group_by() %>% summarize()
```

# `group_by()` `%>%` `summarize()`

Make summaries of your data *by group*



# group\_by() %>% summarize()

- Make summaries of your data

```
covid_testing %>%  
  summarize(order_count = n())
```

mrn	pan_day	order_count
5001412	4	4
5000533	7	
5009134	7	
5008518	8	



# group\_by() %>% summarize()

- Make summaries of your data

```
covid_testing %>%  
  group_by(pan_day) %>%  
  summarize(order_count = n())
```

mrn	pan_day
5001412	4
5000533	7
5009134	7
5008518	8



pan_day	order_count
4	1
7	2
8	3
9	9

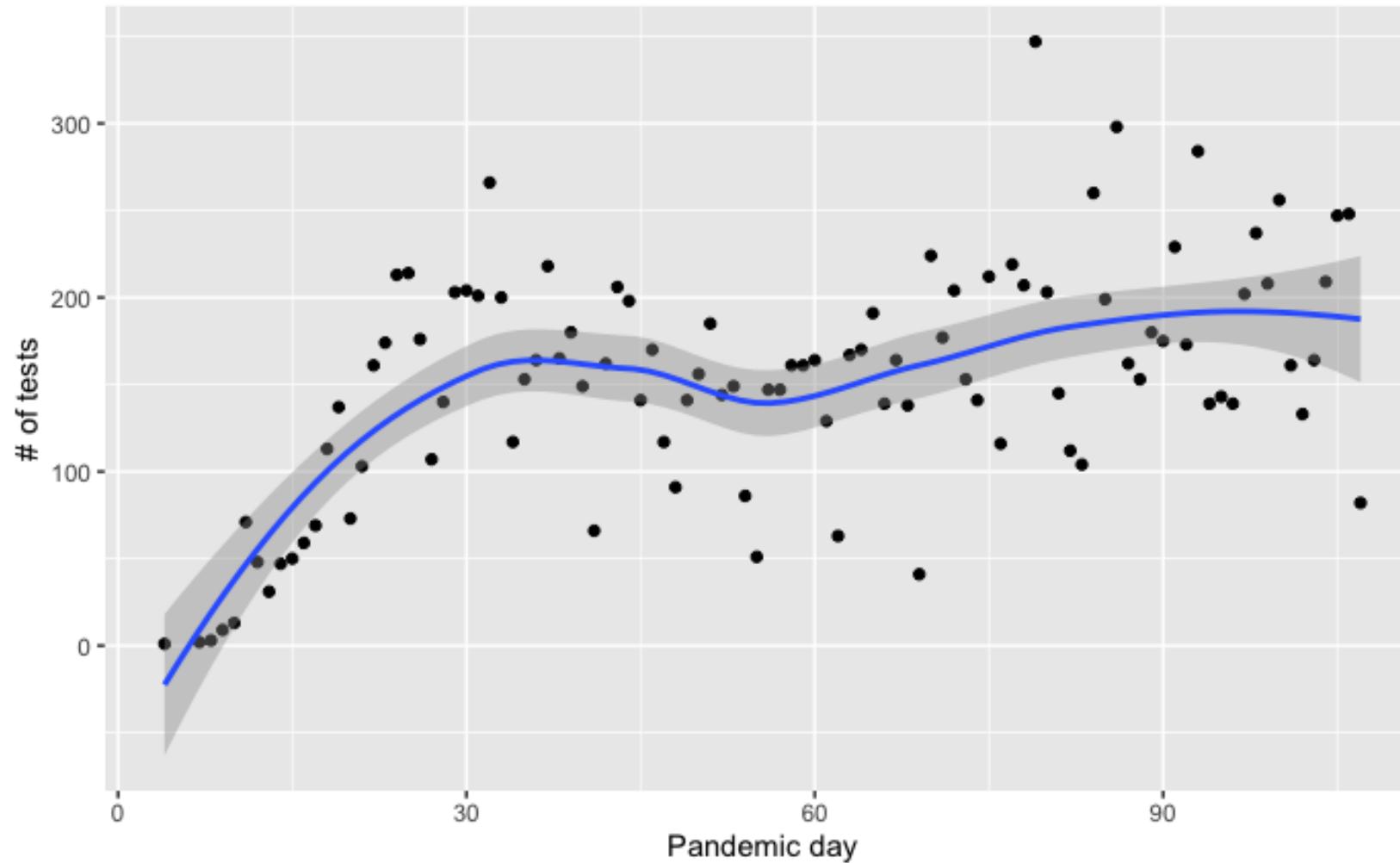


# Your Turn 3

Calculate:

- a) The median turnaround time for each day
- c) (\*Extra\*) The median number of orders per day

# group\_by() %>% summarize(): Example



# Stats: Tests for association

# Q: Is there an association between insurance and SARS-CoV-2 RT-PCR positivity?

payor_group_fac <chr>	negative <int>	positive <int>
commercial	3549	86
government	3318	242
other	309	17
unassigned	7182	520

4 rows



```
data %>%  
  fisher.test(simulate.p.value = T)
```



Fisher's Exact Test for Count Data with simulated p-value (based on 2000 replicates)

```
data: .  
p-value = 0.0004998  
alternative hypothesis: two.sided
```

# Data wrangling - 1

function that flexibly  
assigns values

```
covid_testing_2 <- covid_testing %>%  
  mutate(payor_group_fac = case_when(  
    is.na(payor_group) ~ "unassigned",  
    payor_group %in% c("charity care",  
      "medical assistance",  
      "self pay",  
      "other") ~ "other",  
    TRUE ~ payor_group))  
  ) %>%  
  filter(result %in% c("positive", "negative"))
```



# Data wrangling - 2

```
# Generate counts
tmp_table_tall <- covid_testing_2 %>%
  group_by(payor_group_fac, result) %>%
  summarize(n = n()) %>%
  ungroup()
tmp_table_tall

# Pivot from tall to wide table
tmp_table_wide <- tmp_table_tall %>%
  spread(key = "result", value = "n") %>%
  select(-payor_group_fac)
tmp_table_wide
```

Remove groupings

Maps key values to separate columns



# Testing for association

payor_group_fac <chr>	negative <int>	positive <int>
commercial	3549	86
government	3318	242
other	309	17
unassigned	7182	520

4 rows



```
data %>%  
  fisher.test(simulate.p.value = T)
```



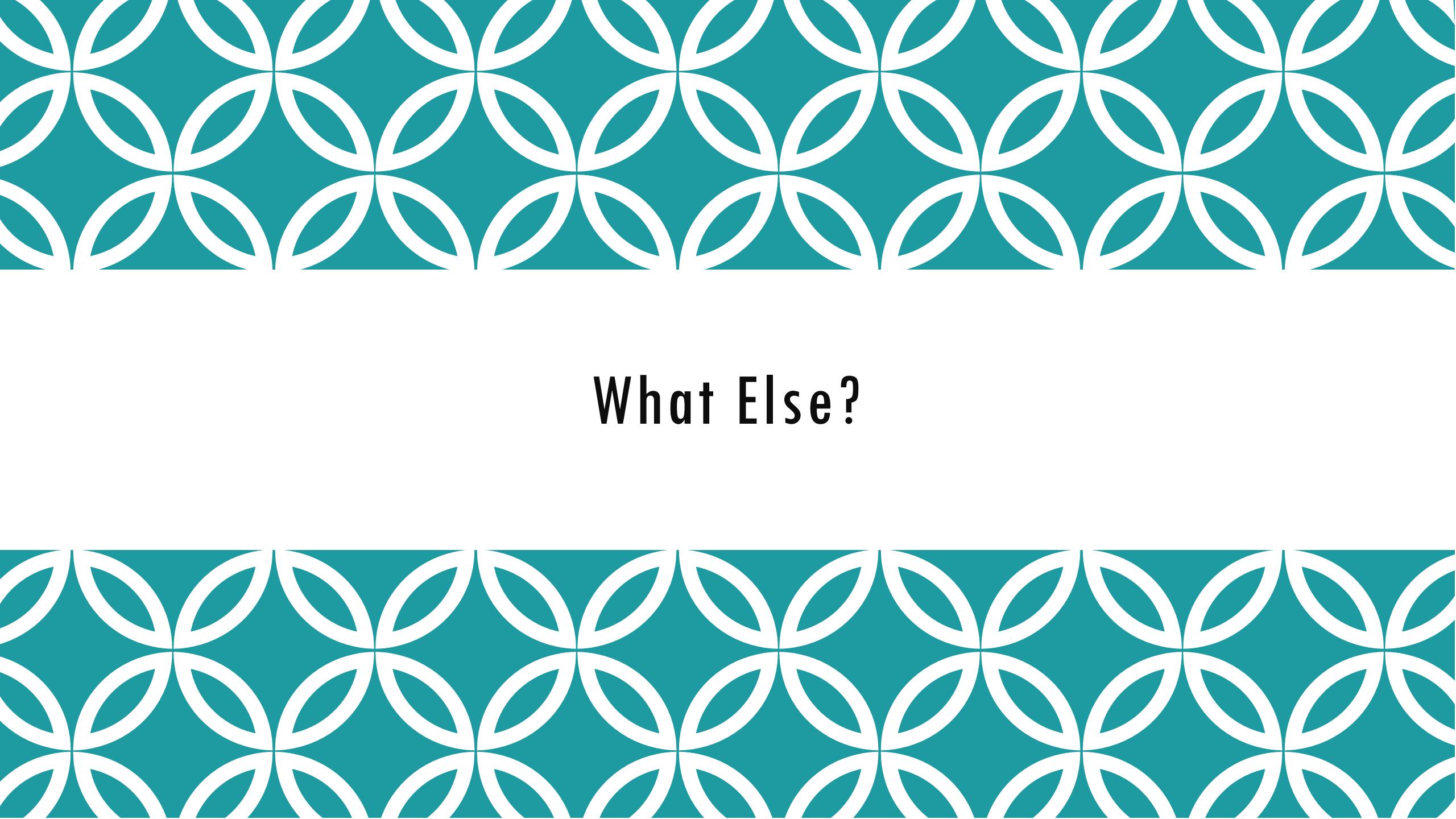
Fisher's Exact Test for Count Data with simulated p-value (based on 2000 replicates)

```
data: .  
p-value = 0.0004998  
alternative hypothesis: two.sided
```



## Your Turn 4

Use `fisher.test()` to estimate the relative odds of a positive test result for patients with government or commercial insurance?



What Else?

# Logistic regression

```
tmp_fit <- tmp %>%  
  glm(result_fac ~ payor_group_fac + age, # model formula  
       data=.,                            # dataset  
       family="binomial")                 # type of model  
  
summary(tmp_fit)
```



# Goals

1. Learn how to summarize and assess data

# Objectives

1. Calculate a summary statistic for a variable using `summarize`
2. Calculate of summary statistic for a variable separately for a group of observations, using `group\_by` and `summarize`
3. Perform a simple test for association