



Statistical Analyses in R

Session 5

Dan Herman

July 16, 2021

July 15, 2021		Session	Instructor
1:00 pm - 1:30 pm		Instructor Introductions, Introduction to technology	Amrom Obstfeld
1:35 pm - 2:05 pm		Introduction to R and RStudio	Joe Rudolf
2:20 pm - 3:15 pm		Reproducible Reporting	Joe Rudolf
3:30 pm - 5:00 pm		Data Visualization	Stephan <u>Kadauke</u>
July 16, 2021			
1:00 pm - 2:30 pm		Data Transformation	Amrom Obstfeld
2:45 pm - 4:15 pm		Statistical Analysis	Dan Herman
4:30 pm - 5:00 pm		Advanced Reporting	Patrick Mathias

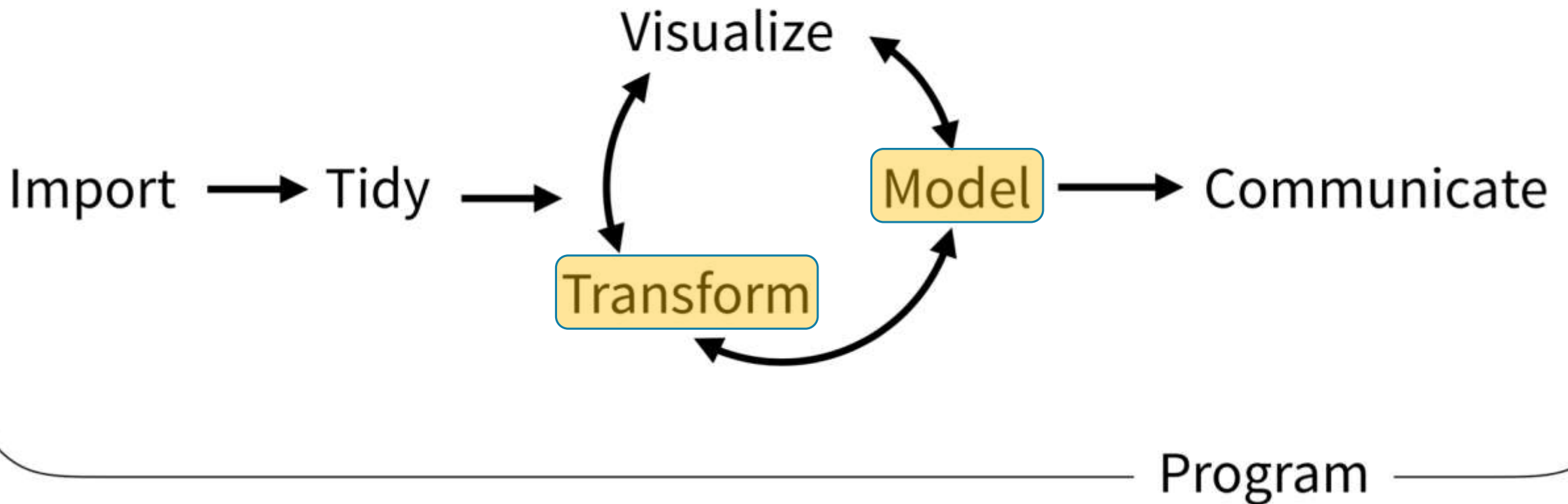
Goals

1. Learn how to summarize data and assess hypotheses

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



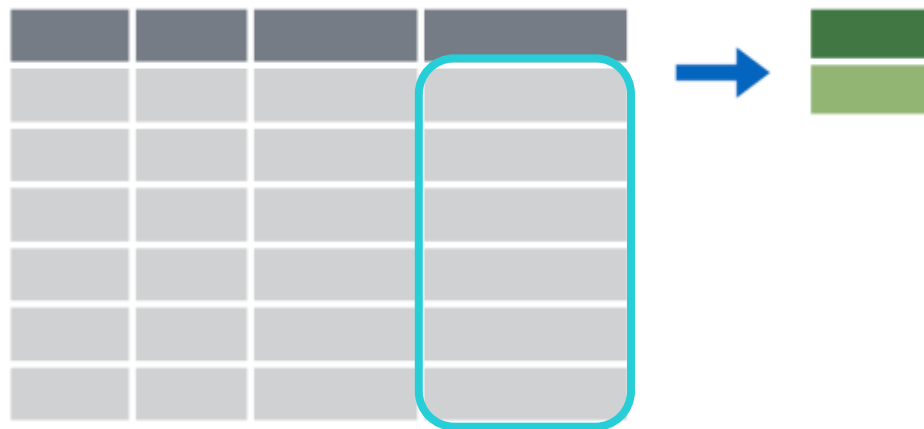


Summarize()



summarize()

- Make summaries of your data



summarize()

- Make summaries of your data

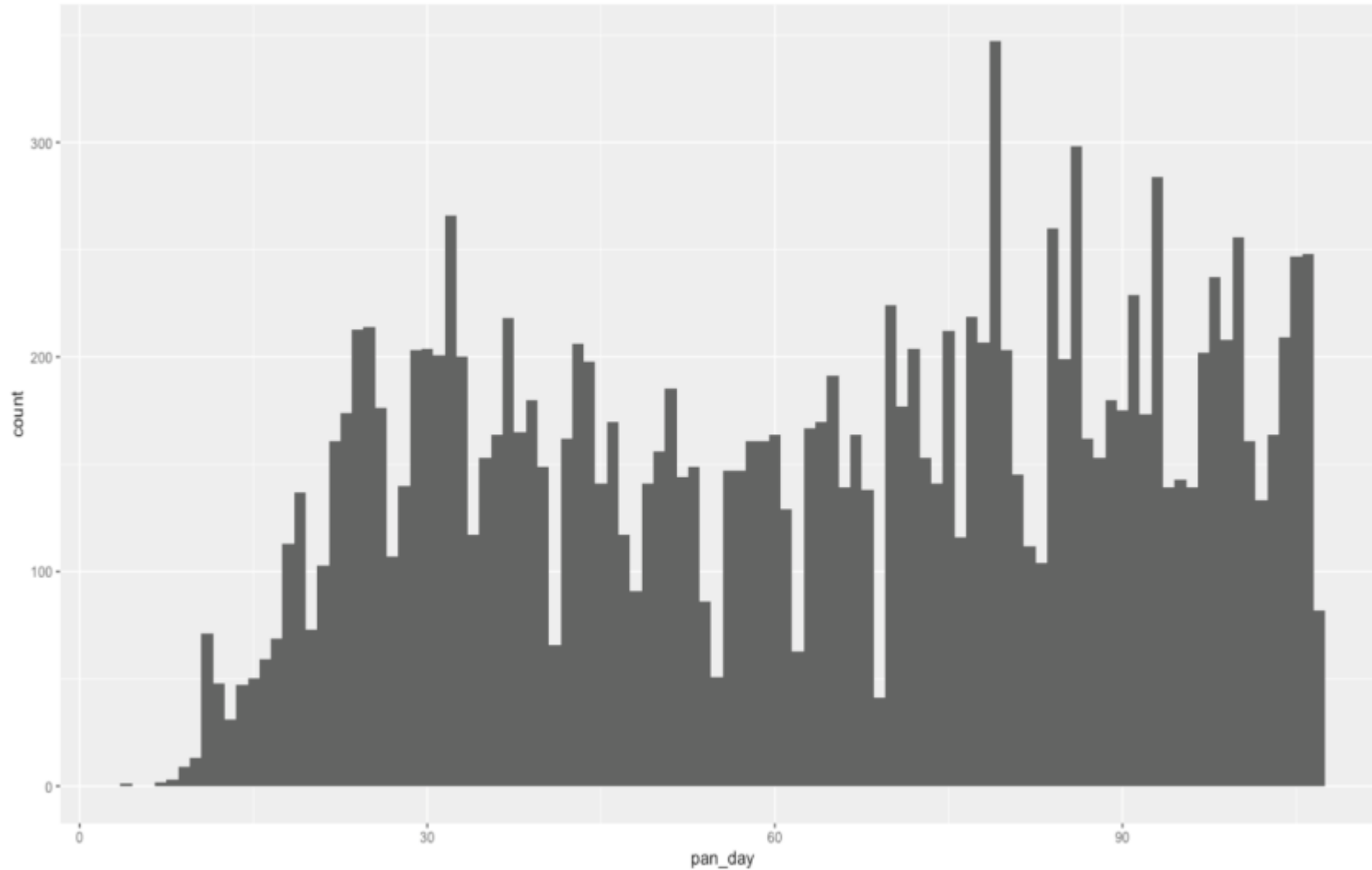
```
covid_testing %>%  
  summarize(new_variable = calculation)
```

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

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.

left_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
 Join matching values from y to x.

right_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
 Join matching values from x to y.

inner_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
 Join data. Retain only rows with matches.

full_join(x, y, by = NULL, copy = FALSE, suffix = c(".x", ".y"), ...)
 Join data. Retain all values, all rows.

Use **by = c("col1", "col2")** to specify the column(s) to match on.
`left_join(x, y, by = "A")`

Use a named vector, **by = c("col1" = "col2")**, to match on columns with different names in each data set.
`left_join(x, y, by = c("C" = "D"))`

Use **suffix** to specify suffix to give to duplicate column names.
`left_join(x, y, by = c("C" = "D"), suffix = c("1", "2"))`

COMBINE CASES

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

Use **bind_rows(..., id = NULL)** Returns tables one on top of the other as a single table. Set `id` to a column name to add a column of the original table names (as pictured)

intersect(x, y, ...)
 Rows that appear in both x and y.

setdiff(x, y, ...)
 Rows that appear in x but not y.

union(x, y, ...)
 Rows that appear in x or y.
 (Duplicates removed). `union_all()` retains duplicates.

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

EXTRACT ROWS

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

semi_join(x, y, by = NULL, ...)
 Return rows of x that have a match in y. USEFUL TO SEE WHAT WILL BE JOINED.

anti_join(x, y, by = NULL, ...)
 Return rows of x that do not have a match in y. USEFUL TO SEE WHAT WILL NOT BE JOINED.



summarize() examples

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



Your Turn 2

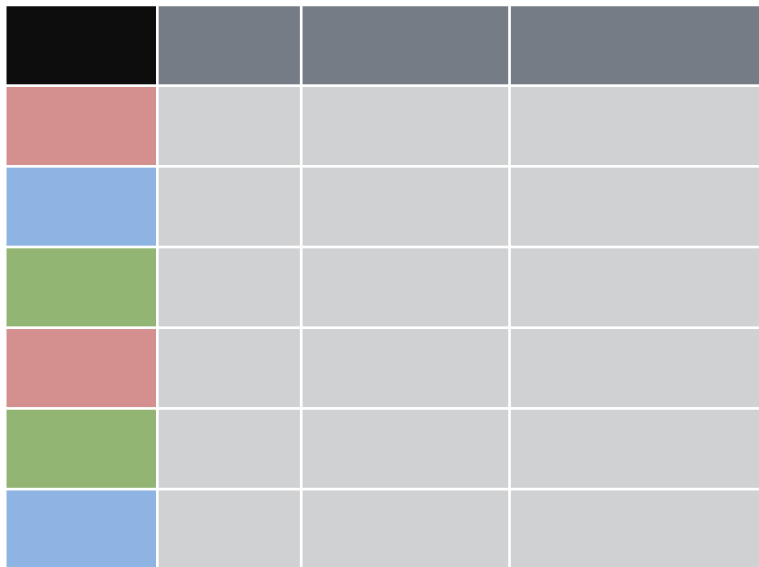
Consider:

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)
```

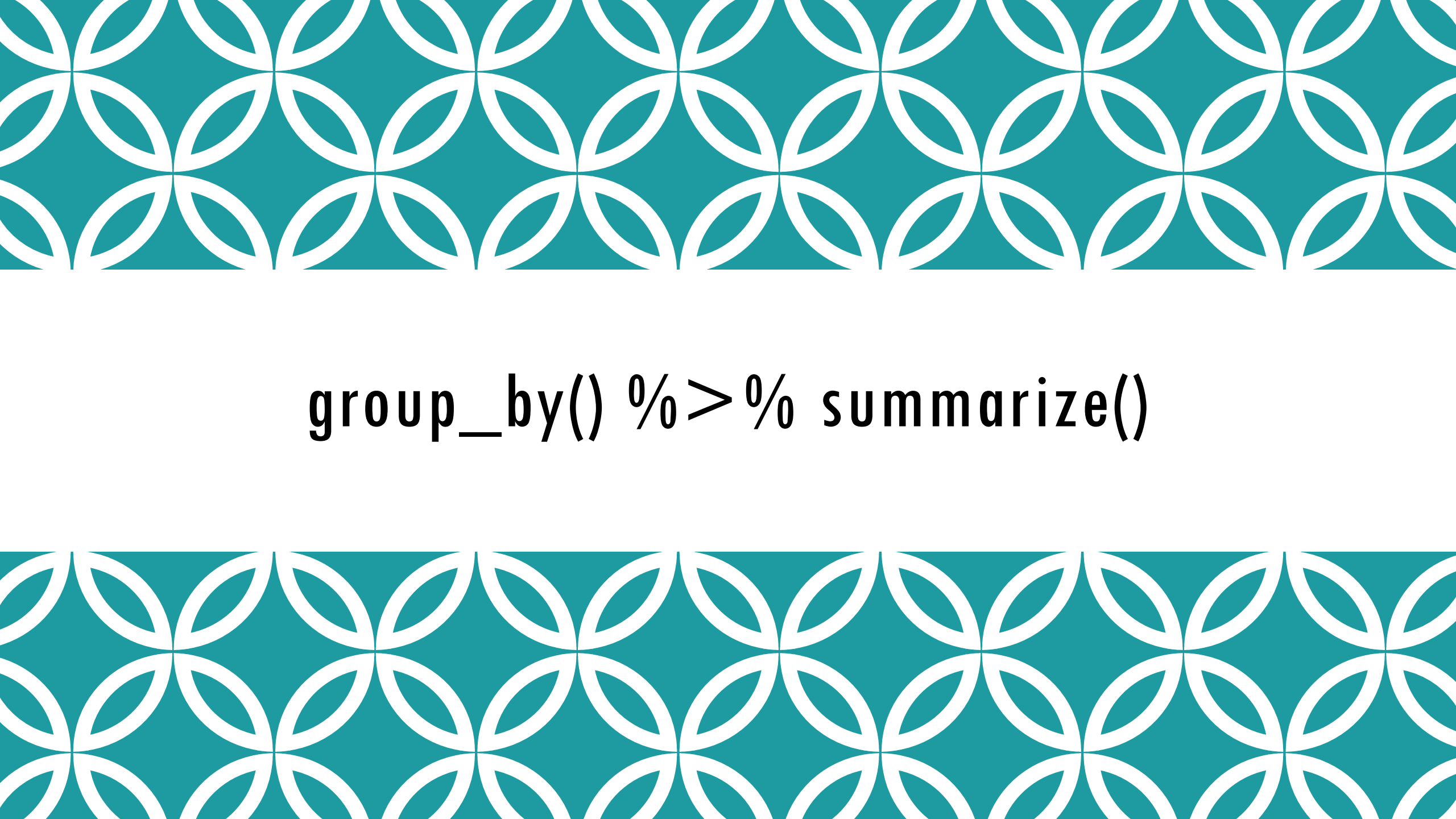
```
# A tibble: 15,524 x 17  
# Groups:   pan_day [102]  
  mrn first_name last_name gender pan_day  
  <dbl> <chr>      <chr>    <chr>    <dbl>  
1 5.00e6 jhezane    westerli... female     4  
2 5.00e6 penny      targaryen female     7  
3 5.01e6 grunt      rivers    male      7  
4 5.01e6 melisandre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

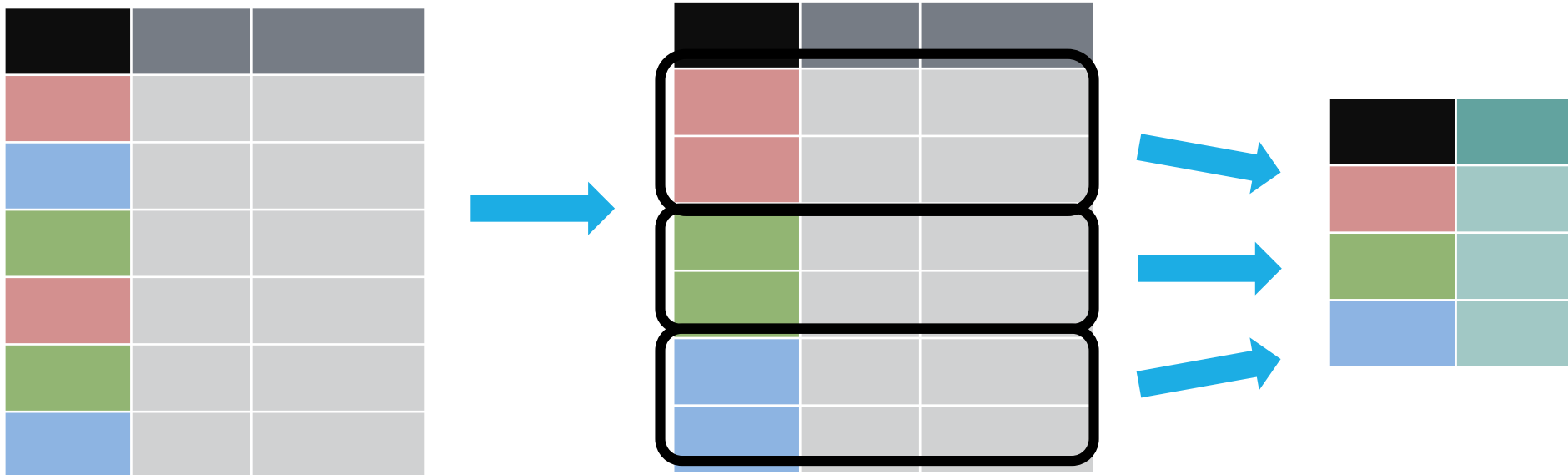
```
# A tibble: 15,524 x 3  
# Groups:   pan_day, clinic_name [2,526]  
  mrn pan_day clinic_name  
  <dbl> <dbl> <chr>  
1 5001412      4 inpatient ward a  
2 5000533      7 clinical lab  
3 5009134      7 clinical lab  
4 5008518      8 clinical lab  
5 5008967      8 emergency dept
```



```
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
5001412	4
5000533	7
5009134	7
5008518	8



order_count
15524



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

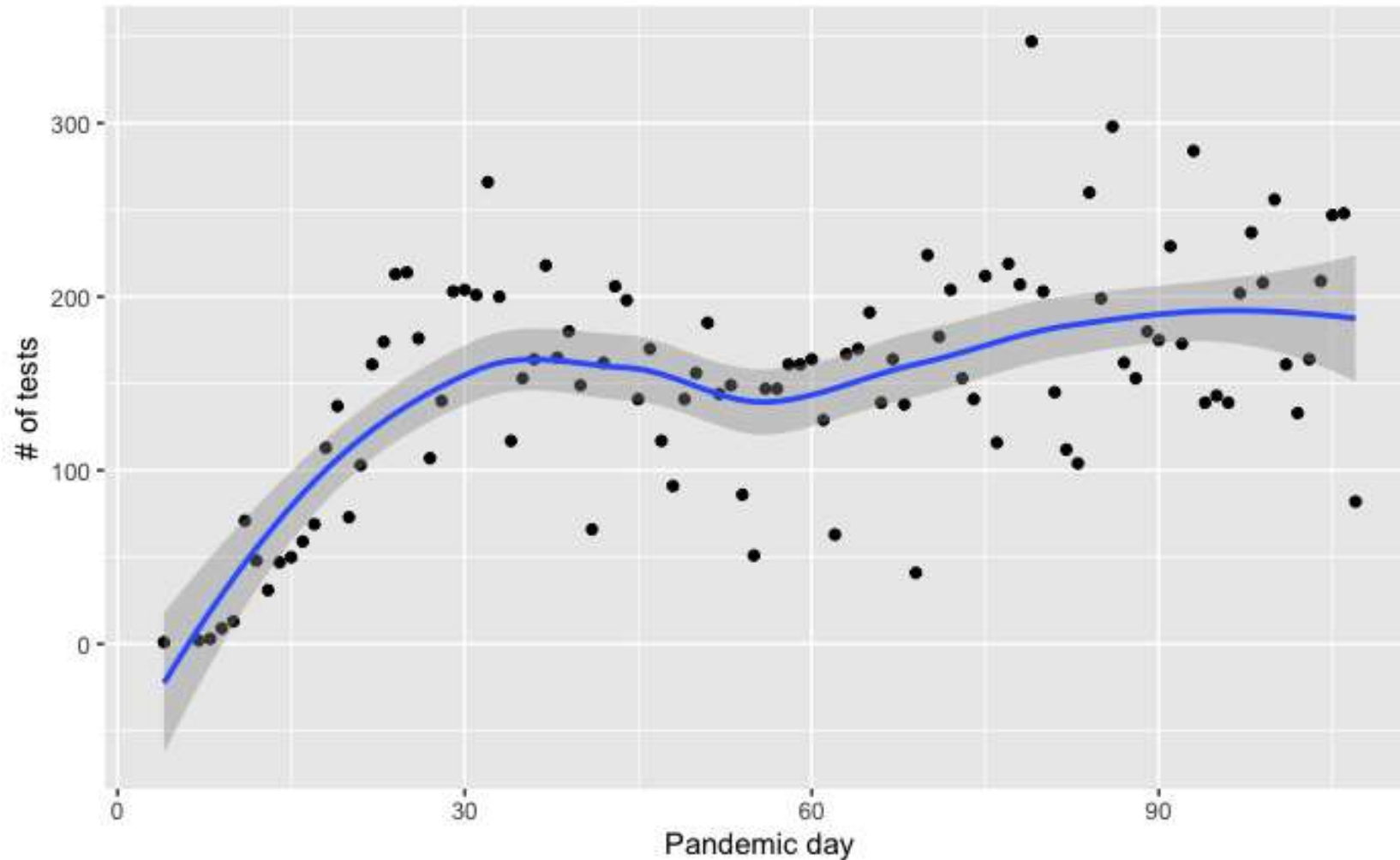


Your Turn 3

Calculate:

- a) The median turnaround time for each day
- b) (**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)
```

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
```

Remove groupings

```
# Pivot from tall to wide table
tmp_table_wide <- tmp_table_tall %>%
  pivot_wider(id_cols = payor_group_fac, names_from = result, values_from = n)
tmp_table_wide
```

Maps result 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 insurance compared to commercial insurance?



What Else?



Logistic regression

```
tmp_fit <- glm(result_fac ~ payor_group_fac + age, # model formula
               data = tmp,                       # 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