

Introduction to R Workshop

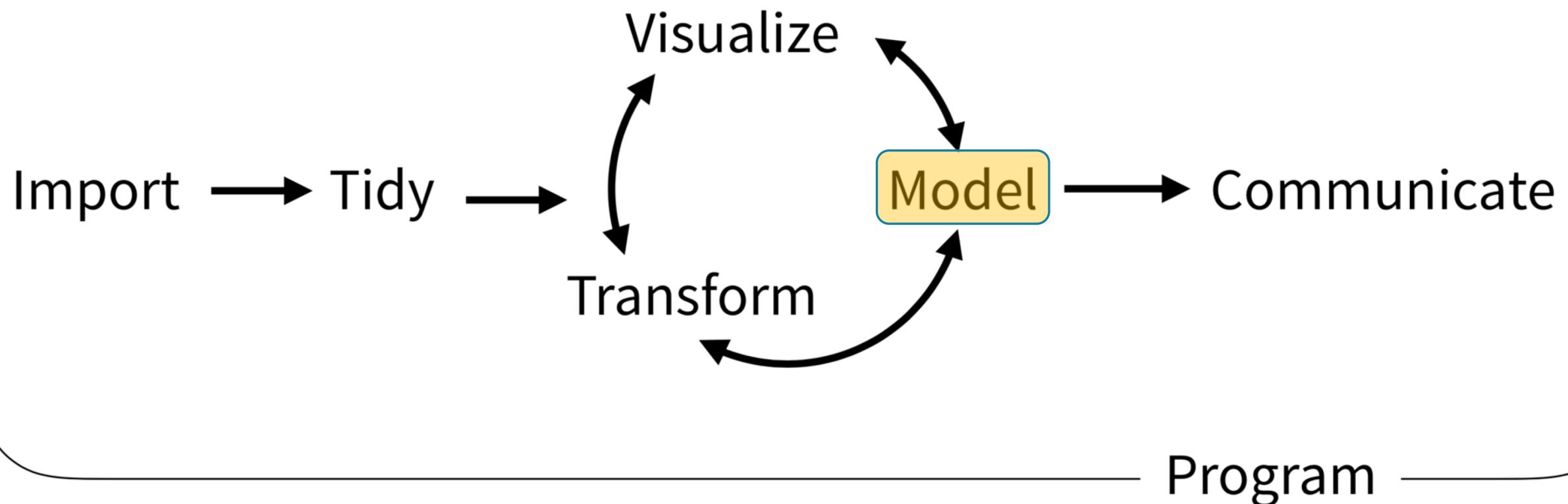
Session 5
Data Understanding: Stats
May 6, 2019

Objectives

Be able to...

1. Summarize variables, including calculation of summary statistics
2. Break apart data frames observations into groups
3. Use standard statistical modeling functions
4. Compare variables' distributions

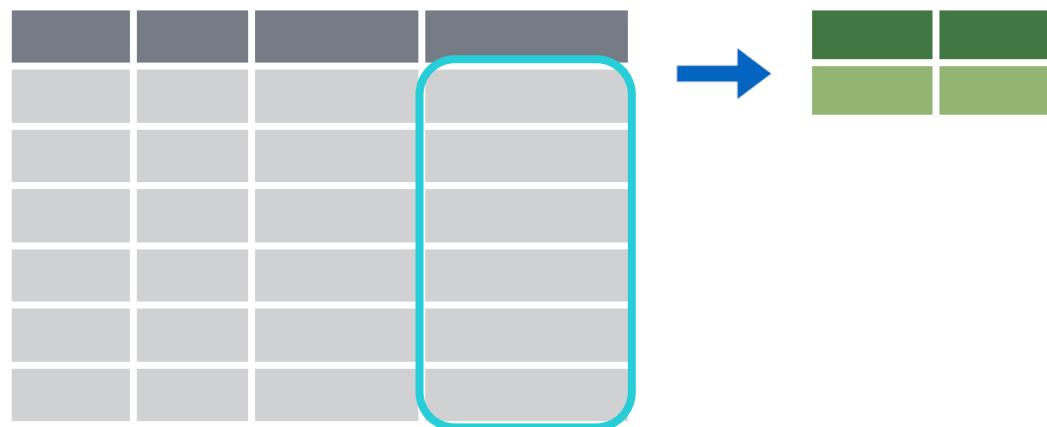
Typical Data Science Pipeline



Summarize()

summarize()

- Make summaries of your data



summarize()

- Make summaries of your data

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

name for new
variable

Value or
function



summarize()

- Make summaries of your data

function that returns
number of observations

```
orders %>%  
  select(order_id, patient_id) %>%  
  head(4) %>%  
  
  summarize(order_count = n())
```

order_id	patient_id
19766	511388
88444	511388
40477	508061
97641	508061



order_count
4



summarize()

- Make summaries of your data

```
orders %>%  
  select(order_id, patient_id) %>%  
  head(4) %>%  
  
  summarize(order_count = n(),  
            pt_count = n_distinct(patient_id))
```

function that returns
number of distinct values

order_id	patient_id
19766	511388
88444	511388
40477	508061
97641	508061



order_count	pt_count
4	2



Your Turn 1

Add onto the code in the above chunk to calculate:

- 1) Mean count of orders per patient
- 2) Mean count of orders per department



Calculate:

- 1) Mean count of orders per patient

```
orders_1 <- orders %>%  
  summarize(order_count = n(),  
            pt_count = n_distinct(patient_id)) %>%  
  mutate(pt_order_count_mean = order_count / pt_count)
```

order_count <int>	pt_count <int>	pt_order_count_mean <dbl>
45002	9406	4.784393
1 row		



Calculate:

2) Mean count of orders per department

```
orders_1 <- orders %>%  
  summarize(order_count = n(),  
            dept_count = n_distinct(department)) %>%  
  mutate(dept_order_count_mean = order_count / dept_count)
```

order_count <int>	dept_count <int>	dept_order_count_mean <dbl>
45002	20	2250.1
1 row		





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 D
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.

`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.

`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

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

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

`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

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

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.

Your Turn 2

Use `summarize()` to calculate:

- 1) The date of the first (or minimum) order
- 2) The median time difference between `order_time` and `result_time`

Hint Refer to help for NA handling



Calculate:

- 1) The minimum or first order date

```
orders %>%  
  summarize(order_time_min = min(order_time))
```

order_time_min
<S3: POSIXct>

2017-08-13 11:59:00

1 row



Calculate:

2) The median time difference between order_time and result_time

```
orders %>%
  mutate(result_interval = result_time - order_time) %>%
  summarize(result_interval_med = median(result_interval, na.rm = TRUE))
```

result_interval_med
<time>
18360 secs
1 row



Your Turn 3

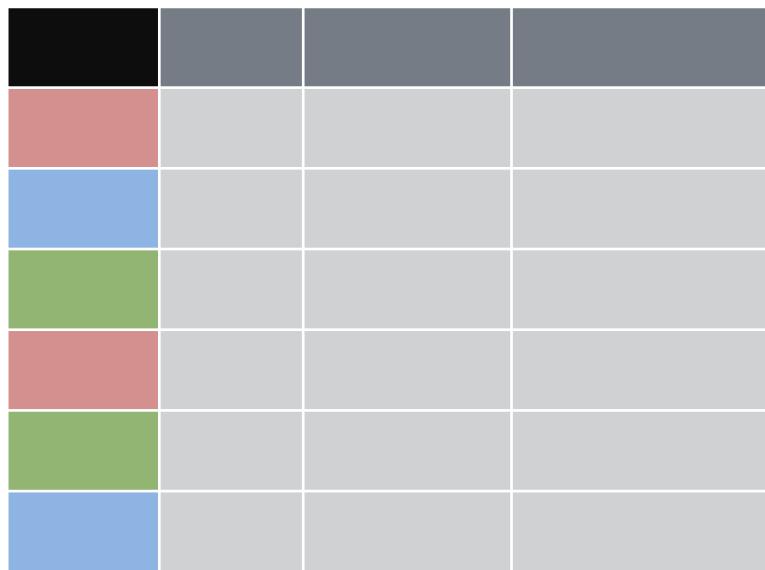
Consider:

How would you calculate the median number of orders for each patient?



`group_by()`

group_by()



group_by()

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

```
orders %>%  
  group_by(variable)
```

name of variable
to group by



group_by()

- *Group observations by patient_id*

```
orders %>%  
  group_by(patient_id)
```

```
# A tibble: 45,002 x 17  
# Groups:   patient_id [9,406]  
  order_id patient_id description proc_code  
    <dbl>      <dbl> <chr>       <chr>  
1 19766      511388 PROTHROMBI... PRO  
2 88444      511388 BASIC META... BMP  
3 40477      508061 THYROID ST... TSH  
4 97641      508061 T4, FREE    T4FR
```



group_by()

- *Group observations by patient_id and department*

```
orders %>%  
  group_by(patient_id, department)
```

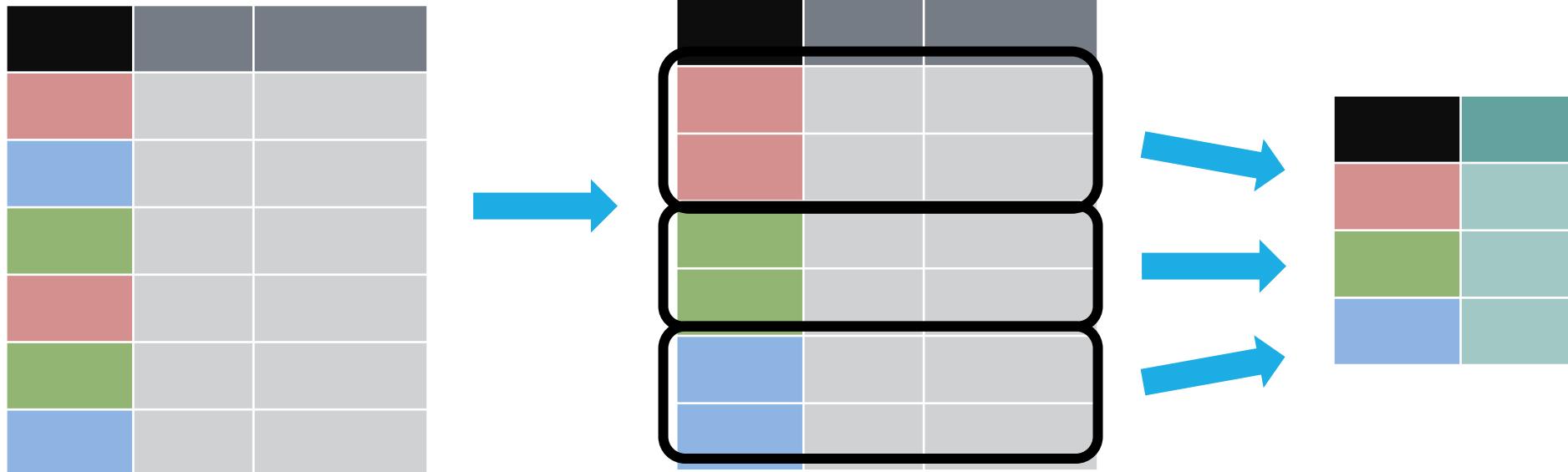
```
# A tibble: 45,002 x 17  
# Groups: patient_id, department [10,102]  
  order_id patient_id description proc_code  
    <dbl>      <dbl> <chr>       <chr>  
1 19766      511388 PROTHROMBI... PRO  
2 88444      511388 BASIC META... BMP  
3 40477      508061 THYROID ST... TSH  
4 97641      508061 T4, FREE   T4FR
```



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

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

Make summaries of your data *by group*



Re-calculate the mean order count per patient

- *Recall that we previously used summarize() and mutate() like this:*

```
orders %>%  
  summarize(order_count = n(),  
            pt_count = n_distinct(patient_id)) %>%  
  mutate(pt_order_count_mean = order_count / pt_count)
```

```
# A tibble: 1 x 3  
  order_count pt_count pt_order_count_mean  
        <int>     <int>             <dbl>  
1         45002      9406            4.784393
```



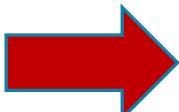
Calculate the mean order count per patient

- Let's now use `group_by()` and `summarize()`...

```
orders %>%  
  group_by(patient_id) %>%  
  summarize(order_count = n()) %>%  
  summarize(pt_order_count_mean = mean(order_count))
```

A tibble: 9,406 x 2
 patient_id order_count

	patient_id	order_count
	<dbl>	<int>
1	500001	1
2	500002	7
3	500003	5
4	500005	1
5	500006	5



2

A tibble: 1 x 1
 pt_order_count_mean
 <dbl>
1 4.784393



Your Turn 4

Calculate:

- 1) The median number of orders per patient
- 2) The maximum number of TSH orders per patient
- 3) (*Bonus*) The 5th and 95th percentile of the number of orders per patient (*Hint: look up function `quantile()``*)



Calculate:

- 1) The median number of orders per patient

```
orders %>%
  group_by(patient_id) %>%
  summarize(order_count = n()) %>%
  summarize(pt_order_count_median = median(order_count))
```

pt_order_count_median
<dbl>
4
1 row



Calculate:

2) The maximum number of TSH orders per patient

```
orders %>%
  filter(proc_code == "TSH") %>%
  group_by(patient_id) %>%
  summarize(order_count = n()) %>%
  summarize(pt_order_count_max = max(order_count))
```

pt_order_count_max
<dbl>
6

1 row



Calculate:

3) (*Bonus*) The 5th and 95th percentile of the number of orders per patient

```
orders %>%  
  group_by(patient_id) %>%  
  summarize(order_count = n()) %>%  
  summarize(pt_order_count_q05 = quantile(order_count, probs=0.05),  
           pt_order_count_q95 = quantile(order_count, probs=0.95))
```

pt_order_count_q05 <dbl>	pt_order_count_q95 <dbl>
1	13

1 row



Your Turn 5

Calculate the mean order count per patient for each department

Hint: summarize() rolls up a single grouping variable at a time

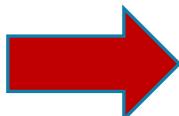


Calculate the mean order count per patient for each department

```
orders %>%
  group_by(department, patient_id) %>%
  summarize(order_count = n()) %>%
  summarize(pt_order_count_mean = mean(order_count))
```

A tibble: 10,102 x 3
 # Groups: department [?]

	department	patient_id	order_count
	<chr>	<dbl>	<int>
1	BEHAVIORAL HEALTH CLINIC	500126	1
2	BEHAVIORAL HEALTH CLINIC	500251	9
3	BEHAVIORAL HEALTH CLINIC	500385	13
4	BEHAVIORAL HEALTH CLINIC	500395	1
5	BEHAVIORAL HEALTH CLINIC	500519	1



A tibble: 20 x 2

	department	pt_order_count_mean
	<chr>	<dbl>
1	BEHAVIORAL HEALTH CLINIC	5.651685
2	CARDIOLOGY CLINIC	3.916031
3	ENDOCRINOLOGY CLINIC	3.416092
4	FAMILY MEDICINE CLINIC	3.609278
5	GASTROENTEROLOGY CLINIC	4.540698

1
2



Compare two sets of count data

Compare sample distributions

- Create the order counts per patient for each department*

```
orders %>%
  filter(department %in% c("INTERNAL MEDICINE CLINIC",
                            "FAMILY MEDICINE CLINIC")) %>%
  group_by(department, patient_id) %>%
  summarize(order_count = n()) %>%
  ungroup() -> orders_per_pt_dept
```

A tibble: 9,831 x 17
Groups: department, patient_id [2,632]
 order_id patient_id description
 <dbl> <dbl>
1 19766 511388 PROTHROMBIN
2 88444 511388 BASIC METABOLIC PANEL
3 50728 501184 COMPREHENSIVE METABOLIC PANEL
4 91635 501184 CBC (HEMOGRAM)



A tibble: 2,632 x 3
Groups: department [?]
 department patient_id order_count
 <chr> <dbl> <int>
1 FAMILY MEDICINE CLINIC 500003 5
2 FAMILY MEDICINE CLINIC 500022 4
3 FAMILY MEDICINE CLINIC 500024 14
4 FAMILY MEDICINE CLINIC 500034 2
5 FAMILY MEDICINE CLINIC 500042 4

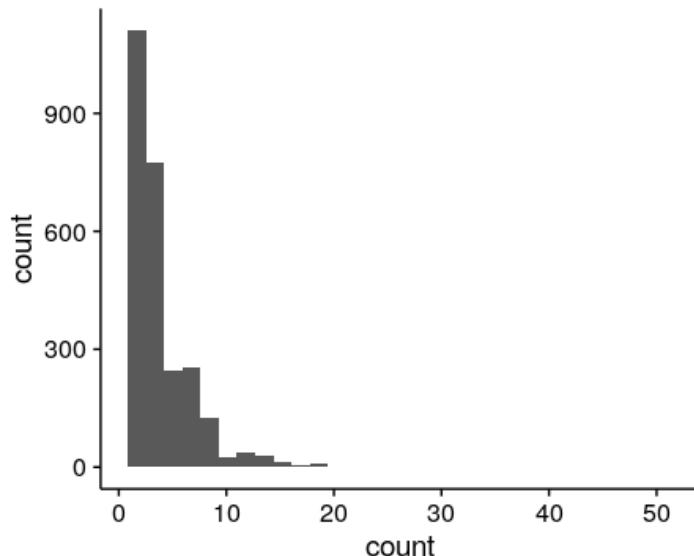


A tibble: 2,632 x 3
 department patient_id order_count
 <chr> <dbl> <int>
1 FAMILY MEDICINE CLINIC 500003 5
2 FAMILY MEDICINE CLINIC 500022 4
3 FAMILY MEDICINE CLINIC 500024 14
4 FAMILY MEDICINE CLINIC 500034 2
5 FAMILY MEDICINE CLINIC 500042 4



Visualize the distribution of order counts

```
orders_per_pt_dept %>%  
  ggplot() +  
  geom_histogram(aes(x = order_count))
```



orders_per_pt_dept

department	patient_id	order_count
<chr>	<dbl>	<int>
FAMILY MEDICINE CLINIC	500003	5
FAMILY MEDICINE CLINIC	500022	4
FAMILY MEDICINE CLINIC	500024	14
FAMILY MEDICINE CLINIC	500034	2
FAMILY MEDICINE CLINIC	500042	4
FAMILY MEDICINE CLINIC	500043	9

6 rows



Describe the central tendency of the order count

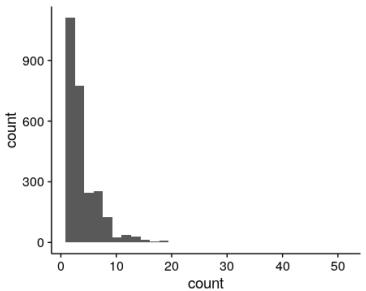
```
orders_per_pt_dept %>%  
  group_by(department) %>%  
  summarize(order_count_median = median(order_count),  
            order_count_mean = mean(order_count))
```

```
# A tibble: 2 x 3  
  department order_count_median order_count_mean  
  <chr>           <dbl>             <dbl>  
1 FAMILY MEDICINE CLINIC      3        3.609278  
2 INTERNAL MEDICINE CLINIC    3        3.808664
```

orders_per_pt_dept

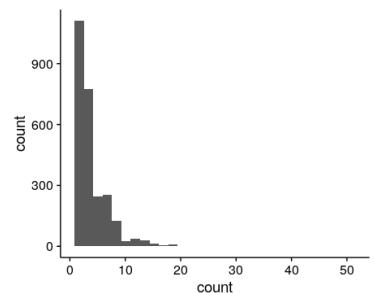
department	patient_id	order_count
<chr>	<dbl>	<int>
FAMILY MEDICINE CLINIC	500003	5
FAMILY MEDICINE CLINIC	500022	4
FAMILY MEDICINE CLINIC	500024	14
FAMILY MEDICINE CLINIC	500034	2
FAMILY MEDICINE CLINIC	500042	4
FAMILY MEDICINE CLINIC	500043	9

6 rows



What test can compare count distributions?

Populations	Parametric	Non-parametric
Two populations	t-test	Mann-Whitney U
Many populations	ANOVA	Kruskal Wallis / one-way anova
Populations across several treatments/times	repeated measures ANOVA	Friedman test



wilcox.test()

dependent or
outcome
variable

tilde
operator

grouping or
predictor
variable

```
wilcox.test(measure ~ group,  
            data = orders, ...)
```

data frame

additional
arguments

Compare distributions by rank order



Compare order count distributions between 2 departments

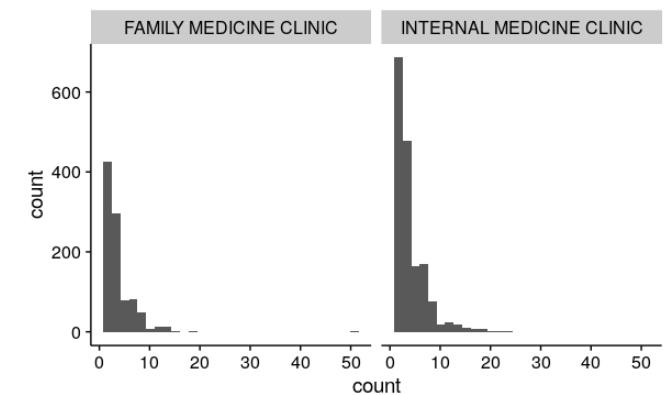
- *Wilcoxon rank-sum test*

```
wilcox.test(order_count ~ department,
            data = orders_per_pt_dept,
            alternative = "two.sided",
            paired = FALSE,
            conf.int = TRUE)
```

Wilcoxon rank sum test with continuity correction

```
data: count by department
W = 779510, p-value = 0.1521
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
-3.210939e-06 1.571466e-05
sample estimates:
difference in location
-4.511812e-05
```

```
# A tibble: 6 x 3
  department patient_id count
  <chr>        <dbl> <int>
1 FAMILY MEDICINE CLINIC 500003    5
2 FAMILY MEDICINE CLINIC 500022    4
3 FAMILY MEDICINE CLINIC 500024   14
4 FAMILY MEDICINE CLINIC 500034    2
5 FAMILY MEDICINE CLINIC 500042    4
6 FAMILY MEDICINE CLINIC 500043    9
```



Your Turn 6

Compare order counts across the nephrology, cardiology, and gastroenterology departments.

**Hint: Convert department to a factor using function factor()*



Compare order counts across the nephrology, cardiology, and gastroenterology departments

```
orders %>%
  mutate(department = factor(department)) %>%
  filter(department %in% c("NEPHROLOGY CLINIC",
                            "CARDIOLOGY CLINIC",
                            "GASTROENTEROLOGY CLINIC")) %>%
  group_by(department, patient_id) %>%
  summarize(order_count = n()) %>%
  ungroup() %>%
  kruskal.test(order_count ~ department,
               data = .)
```

Kruskal-Wallis rank sum test

```
data: order_count by department
Kruskal-Wallis chi-squared = 6.3171, df = 2, p-value = 0.04249
```

