

Intro to R

Data Summarization

Part 1: Numeric / continuous data

Data Summarization

- Basic statistical summarization
 - `mean(x)`: takes the mean of `x`
 - `sd(x)`: takes the standard deviation of `x`
 - `median(x)`: takes the median of `x`
 - `quantile(x)`: displays sample quantiles of `x`. Default is min, IQR, max
 - `range(x)`: displays the range. Same as `c(min(x), max(x))`
 - `sum(x)`: sum of `x`
 - `max(x)`: maximum value in `x`
 - `min(x)`: minimum value in `x`
- **all have the `na.rm` = argument for missing data**

Statistical summarization

The vector getting summarized goes inside the parentheses:

```
x <- c(1, 5, 7, 4, 2, 8)  
mean(x)
```

```
[1] 4.5
```

```
range(x)
```

```
[1] 1 8
```

```
sum(x)
```

```
[1] 27
```

Statistical summarization

Note that many of these functions have additional inputs regarding missing data, typically requiring the `na.rm` argument (“remove NAs”).

```
x <- c(1, 5, 7, 4, 2, 8, NA)
mean(x)
```

```
[1] NA
```

```
mean(x, na.rm = TRUE)
```

```
[1] 4.5
```

```
quantile(x)
```

```
Error in quantile.default(x): missing values and NaN's not allowed if 'na.rm' is FALSE
```

```
quantile(x, na.rm = TRUE)
```

0%	25%	50%	75%	100%
1.0	2.5	4.5	6.5	8.0

Statistical summarization

You can only do summarization on numeric or logical types. Not characters.

```
x <- c(1, 5, 7, 4, 2, 8)
sum(x)
```

```
[1] 27
```

```
z <- c("hello", "goodbye")
sum(z)
```

```
Error in sum(z): invalid 'type' (character) of argument
```

But how do we do this on dataframes?

First we will need to learn about something called the “pipe”.

The pipe is this operator in R:

`%>%`

It tells R to “pipe” the dataset on the left into the next function.

Using the pipe %>%

```
states <- read_csv("https://hutchdatascience.org/SeattleStatSummer_R/data/states.csv")
states %>% head() # Same as head(states)!
```

```
# A tibble: 6 × 14
```

	entity	state_abb	state_area_sq_mil...	state_division	state_region	population
	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>
1	Alabama	AL	51609	East South Ce...	South	4903185
2	Alaska	AK	589757	Pacific	West	731545
3	Arizona	AZ	113909	Mountain	West	7278717
4	Arkansas	AR	53104	West South Ce...	South	3017804
5	California	CA	158693	Pacific	West	39512223
6	Colorado	CO	104247	Mountain	West	5758736

```
# ... with 8 more variables: births_in_2021 <dbl>, fertility_rate_per_1000 <dbl>,  
#   cesarean_percent <dbl>, life_expect <dbl>, cancer_rate_per_100000 <dbl>,  
#   cancer_mortality <dbl>, Administered_Dose1_Pop_Pct <dbl>,  
#   Series_Complete_Pop_Pct <dbl>
```


States data

`colnames()` will show us the column names.

```
colnames(states)
```

```
[1] "entity"           "state_abb"  
[3] "state_area_sq_miles" "state_division"  
[5] "state_region"     "population"  
[7] "births_in_2021"   "fertility_rate_per_1000"  
[9] "cesarean_percent" "life_expect"  
[11] "cancer_rate_per_100000" "cancer_mortality"  
[13] "Administered_Dose1_Pop_Pct" "Series_Complete_Pop_Pct"
```

States data

We can also use the pipe:

```
states %>% colnames()
```

```
[1] "entity" "state_abb"  
[3] "state_area_sq_miles" "state_division"  
[5] "state_region" "population"  
[7] "births_in_2021" "fertility_rate_per_1000"  
[9] "cesarean_percent" "life_expect"  
[11] "cancer_rate_per_100000" "cancer_mortality"  
[13] "Administered_Dose1_Pop_Pct" "Series_Complete_Pop_Pct"
```

Summarizing the data

Summarize the data: **summarize()** function

`summarize` creates a summary table of a column you're interested in.

General format - Not the code!

`{data to use} %>%`

`summarize({summary column name} = {operator(source column)})`

Summarize the data: **dplyr** `summarize()` function

`summarize` creates a summary table of a column you're interested in.

General format - Not the code!

```
{data to use} %>%  
  summarize({summary column name} = {operator(source column)})
```

```
states %>%  
  summarize(mean_population = mean(population))
```

```
# A tibble: 1 × 1  
  mean_population  
      <dbl>  
1      6373716.
```

What if there are NAs in my data?

```
states %>%  
  summarize(mean_population = mean(cesarean_percent))
```

```
# A tibble: 1 × 1  
  mean_population  
      <dbl>  
1             NA
```

```
states %>%  
  summarize(mean_population = mean(cesarean_percent, na.rm = TRUE))
```

```
# A tibble: 1 × 1  
  mean_population  
      <dbl>  
1          30.9
```

add `na.rm = TRUE`.

Summarize the data: **dplyr** `summarize()` function

`summarize()` can do multiple operations at once. Separate by a comma.
Breaking line between these keeps things tidy!

```
states %>%  
  summarize(mean_population = mean(population),  
            median_population = median(population))
```

```
# A tibble: 1 × 2  
  mean_population median_population  
    <dbl>          <dbl>  
1    6373716.      4342705
```

summary() Function

Using `summary()` can give you rough snapshots of each numeric column (character columns are skipped):

```
summary(states)
```

entity	state_abb	state_area_sq_miles	state_division
Length:52	Length:52	Min. : 68	Length:52
Class :character	Class :character	1st Qu.: 32675	Class :character
Mode :character	Mode :character	Median : 54629	Mode :character
		Mean : 69654	
		3rd Qu.: 82587	
		Max. : 589757	
state_region	population	births_in_2021	fertility_rate_per_1000
Length:52	Min. : 578759	Min. : 5384	Min. : 30.80
Class :character	1st Qu.: 1790876	1st Qu.: 18778	1st Qu.: 53.83
Mode :character	Median : 4342705	Median : 50312	Median : 56.45
	Mean : 6373716	Mean : 70838	Mean : 56.36
	3rd Qu.: 7362761	3rd Qu.: 82266	3rd Qu.: 60.70
	Max. : 39512223	Max. : 420608	Max. : 68.60
cesarean_percent	life_expect	cancer_rate_per_100000	cancer_mortality
Min. : 23.40	Min. : 71.90	Min. : 121.0	Min. : 1093
1st Qu.: 28.62	1st Qu.: 75.38	1st Qu.: 140.7	1st Qu.: 3514
Median : 31.05	Median : 76.80	Median : 150.8	Median : 8921
Mean : 30.93	Mean : 76.62	Mean : 150.3	Mean : 12085
3rd Qu.: 33.58	3rd Qu.: 78.10	3rd Qu.: 159.2	3rd Qu.: 14356
Max. : 38.50	Max. : 80.70	Max. : 184.7	Max. : 59503
NA's : 2	NA's : 2	NA's : 2	NA's : 2

Let's practice!

Practice

Modify the code below from the `states` dataset to `summarize()` the `fertility_rate_per_1000` column. Find the mean, min, and max.

```
states %>%  
  summarize(____ = mean(____),  
            ____ = min(____),  
            ____ = max(____))
```

Practice

Modify the code below from the `states` dataset to `summarize()` the `fertility_rate_per_1000` column. Find the mean, min, and max.

```
states %>%  
  summarize(mean_fert = mean(fertility_rate_per_1000),  
            min_fert = min(fertility_rate_per_1000),  
            max_fert = max(fertility_rate_per_1000))
```

```
# A tibble: 1 × 3  
  mean_fert min_fert max_fert  
    <dbl>    <dbl>    <dbl>  
1    56.4    30.8    68.6
```

Summary Part 1

- don't forget the `na.rm = TRUE` argument!
- `summary(x)`: quantile information
- `summarize`: creates a summary table of columns of interest

Part 2: Categorical data

count function

Use count to return the number of rows of data.

```
states %>% count()
```

```
# A tibble: 1 × 1  
      n  
  <int>  
1    52
```

count function

Use `count` to return a frequency table of unique elements of a category (column).

```
states %>% count(state_region)
```

```
# A tibble: 5 × 2
  state_region      n
  <chr>         <int>
1 North Central    12
2 Northeast        9
3 South           17
4 West            13
5 <NA>             1
```

count function

Multiple columns listed further subdivides the count.

```
states %>% count(state_region, state_division)
```

```
# A tibble: 10 × 3
```

	state_region	state_division	n
	<chr>	<chr>	<int>
1	North Central	East North Central	5
2	North Central	West North Central	7
3	Northeast	Middle Atlantic	3
4	Northeast	New England	6
5	South	East South Central	4
6	South	South Atlantic	9
7	South	West South Central	4
8	West	Mountain	8
9	West	Pacific	5
10	<NA>	<NA>	1

Grouping

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by variables/columns you specify:

```
# Regular data  
states
```

```
# A tibble: 52 × 14
```

	entity <chr>	state_abb <chr>	state_area_sq_m... <dbl>	state_division <chr>	state_region <chr>	population <dbl>
1	Alabama	AL	51609	East South Ce...	South	4903185
2	Alaska	AK	589757	Pacific	West	731545
3	Arizona	AZ	113909	Mountain	West	7278717
4	Arkansas	AR	53104	West South Ce...	South	3017804
5	California	CA	158693	Pacific	West	39512223
6	Colorado	CO	104247	Mountain	West	5758736
7	Connecticut	CT	5009	New England	Northeast	3565287
8	Delaware	DE	2057	South Atlantic	South	973764
9	Florida	FL	58560	South Atlantic	South	21477737
10	Georgia	GA	58876	South Atlantic	South	10617423

```
# ... with 42 more rows, and 8 more variables: births_in_2021 <dbl>,  
# fertility_rate_per_1000 <dbl>, cesarean_percent <dbl>, life_expect <dbl>,  
# cancer_rate_per_100000 <dbl>, cancer_mortality <dbl>,  
# Administered_Dose1_Pop_Pct <dbl>, Series_Complete_Pop_Pct <dbl>
```

Perform Operations By Groups: dplyr

`group_by` allows you group the data set by variables/columns you specify:

```
states_grouped <- states %>% group_by(state_region)
states_grouped
```

```
# A tibble: 52 × 14
```

```
# Groups:   state_region [5]
```

	entity	state_abb	state_area_sq_m...	state_division	state_region	population
	<chr>	<chr>	<dbl>	<chr>	<chr>	<dbl>
1	Alabama	AL	51609	East South Ce...	South	4903185
2	Alaska	AK	589757	Pacific	West	731545
3	Arizona	AZ	113909	Mountain	West	7278717
4	Arkansas	AR	53104	West South Ce...	South	3017804
5	California	CA	158693	Pacific	West	39512223
6	Colorado	CO	104247	Mountain	West	5758736
7	Connecticut	CT	5009	New England	Northeast	3565287
8	Delaware	DE	2057	South Atlantic	South	973764
9	Florida	FL	58560	South Atlantic	South	21477737
10	Georgia	GA	58876	South Atlantic	South	10617423

```
# ... with 42 more rows, and 8 more variables: births_in_2021 <dbl>,
```

```
#   fertility_rate_per_1000 <dbl>, cesarean_percent <dbl>, life_expect <dbl>,
```

```
#   cancer_rate_per_100000 <dbl>, cancer_mortality <dbl>,
```

```
#   Administered_Dose1_Pop_Pct <dbl>, Series_Complete_Pop_Pct <dbl>
```

Summarize the grouped data

It's grouped! Grouping doesn't change the data in any way, but how **functions operate on it**. Now we can summarize `population` by group:

```
states_grouped %>% summarize(total_population = sum(population))
```

```
# A tibble: 5 × 2
  state_region total_population
  <chr>         <dbl>
1 North Central 68329004
2 Northeast    55982803
3 South        125580448
4 West         78347268
5 <NA>         3193694
```

Use the **pipe** to string these together!

Pipe states into `group_by`, then pipe that into `summarize`:

```
states %>%  
  group_by(state_region) %>%  
  summarize(total_population = sum(population))
```

```
# A tibble: 5 × 2  
  state_region total_population  
  <chr>         <dbl>  
1 North Central    68329004  
2 Northeast       55982803  
3 South          125580448  
4 West            78347268  
5 <NA>           3193694
```

Let's practice!

Practice

Modify the code to group by `state_region` and summarize by average `fertility_rate_per_1000`.

```
states %>%  
  group_by(____) %>%  
  summarize(____ = mean(____))
```

Practice

Modify the code to group by `state_region` and summarize by average `fertility_rate_per_1000`.

```
states %>%  
  group_by(state_region) %>%  
  summarize(avg_fert = mean(fertility_rate_per_1000))
```

```
# A tibble: 5 × 2  
  state_region avg_fert  
  <chr>        <dbl>  
1 North Central 60.1  
2 Northeast    51.2  
3 South        58.0  
4 West         56.3  
5 <NA>         30.8
```


Counting

`n()` can also give you the sample size per group (NAs included).

```
states %>%  
  group_by(state_region) %>%  
  summarize(total_population = sum(population),  
            sample_size = n())
```

```
# A tibble: 5 × 3  
  state_region total_population sample_size  
  <chr>          <dbl>         <int>  
1 North Central    68329004         12  
2 Northeast       55982803          9  
3 South          125580448        17  
4 West           78347268        13  
5 <NA>           3193694          1
```

Summary

- don't forget the `na.rm = TRUE` argument!
- `summary()`: quantile information
- `summarize`: creates a summary table of columns of interest
- `count(x)`: what unique values do you have?
- `group_by(x)`: changes all subsequent functions
 - combine with `summarize()` to get statistics per group
- `summarize()` with `n()` gives the sample size (NAs included)

[Workshop Website](#)