Data Summarization

Data Wrangling in R

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

These functions work on **vectors**:

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

```
[1] 4.5
```

Summarization on a `data.frame`/`tibble`:

```
mtcars %>% pull(hp) %>% mean()
```

```
[1] 146.6875
```

Youth Tobacco Survey

Let's use the Youth Tobacco Survey data again:

```
yts <-
  read_csv("https://sisbid.github.io/Data-Wrangling/data/Youth_Tobacco_Survey_YTS_Data.csv")
head(yts)
```

```
# A tibble: 6 × 31
   YEAR LocationAbbr LocationDesc TopicType     TopicDesc MeasureDesc DataSource
  <dbl> <chr>        <chr>        <chr>         <chr>     <chr>       <chr>
1  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
2  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
3  2015 AZ           Arizona      Tobacco Use … Cessatio… Percent of… YTS
4  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
5  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
6  2015 AZ           Arizona      Tobacco Use … Cessatio… Quit Attem… YTS
# ℹ 24 more variables: Response <chr>, Data_Value_Unit <chr>,
#   Data_Value_Type <chr>, Data_Value <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Data_Value_Std_Err <dbl>,
#   Low_Confidence_Limit <dbl>, High_Confidence_Limit <dbl>, Sample_Size <dbl>,
#   Gender <chr>, Race <chr>, Age <chr>, Education <chr>, GeoLocation <chr>,
#   TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, StratificationID1 <chr>,
#   StratificationID2 <chr>, StratificationID3 <chr>, …
```

Column to vector

Let's work with one column as a vector using `pull()`.

```
locations <- yts %>% pull(LocationDesc)
locations
```

```
  [1] "Arizona"                "Arizona"
  [3] "Arizona"                "Arizona"
  [5] "Arizona"                "Arizona"
  [7] "Arizona"                "Arizona"
  [9] "Arizona"                "Arizona"
 [11] "Arizona"                "Arizona"
 [13] "Arizona"                "Arizona"
 [15] "Arizona"                "Arizona"
 [17] "Arizona"                "Arizona"
 [19] "Arizona"                "Arizona"
 [21] "Arizona"                "Arizona"
 [23] "Arizona"                "Arizona"
 [25] "Connecticut"            "Connecticut"
 [27] "Connecticut"            "Connecticut"
 [29] "Connecticut"            "Connecticut"
 [31] "Connecticut"            "Connecticut"
 [33] "Connecticut"            "Connecticut"
 [35] "Connecticut"            "Connecticut"
 [37] "Connecticut"            "Connecticut"
 [39] "Connecticut"            "Connecticut"
 [41] "Connecticut"            "Connecticut"
 [43] "Connecticut"            "Connecticut"
 [45] "Connecticut"            "Connecticut"
```

Check for NAs

use `sum(is.na())`:

```
sum(is.na(locations))
```

```
[1] 0
```

dplyr: count

Use `count` directly on a data.frame and column: count the number of rows in each group. The `nrow` of the dataset tells you the number of unique groups.

```
yts %>% count(LocationDesc)
```

```
# A tibble: 50 × 2
   LocationDesc             n
   <chr>                <int>
 1 Alabama                378
 2 Arizona                240
 3 Arkansas               210
 4 California              96
 5 Colorado                48
 6 Connecticut            384
 7 Delaware               312
 8 District of Columbia    48
 9 Florida                 96
10 Georgia                282
# ℹ 40 more rows
```

dplyr: count

Multiple columns listed further subdivides the count.

```
yts %>% count(LocationDesc, TopicDesc)
```

```
# A tibble: 146 × 3
   LocationDesc TopicDesc                          n
   <chr>        <chr>                          <int>
 1 Alabama      Cessation (Youth)                 90
 2 Alabama      Cigarette Use (Youth)            144
 3 Alabama      Smokeless Tobacco Use (Youth)    144
 4 Arizona      Cessation (Youth)                 60
 5 Arizona      Cigarette Use (Youth)             99
 6 Arizona      Smokeless Tobacco Use (Youth)     81
 7 Arkansas     Cessation (Youth)                 42
 8 Arkansas     Cigarette Use (Youth)             78
 9 Arkansas     Smokeless Tobacco Use (Youth)     90
10 California   Cessation (Youth)                 24
# ℹ 136 more rows
```

dplyr: count

Option to sort the results with `sort = TRUE`

```
yts %>% count(LocationDesc, sort = TRUE)
```

```
# A tibble: 50 × 2
   LocationDesc       n
   <chr>          <int>
 1 Mississippi      567
 2 New Jersey       387
 3 Connecticut      384
 4 Alabama          378
 5 North Carolina   366
 6 Wisconsin        360
 7 West Virginia    336
 8 North Dakota     330
 9 Pennsylvania     330
10 Oklahoma         318
# ℹ 40 more rows
```

dplyr: count

Instead of counting the number of rows in each group, `wt` computes sum(wt) for each group.

```
# Add up "Data_Value" for each LocationDesc category
yts %>% count(LocationDesc, wt = Data_Value)
```

```
# A tibble: 50 × 2
   LocationDesc            n
   <chr>               <dbl>
 1 Alabama             9220.
 2 Arizona             3937.
 3 Arkansas            5443.
 4 California          2059.
 5 Colorado            1136.
 6 Connecticut         5838.
 7 Delaware            5886 
 8 District of Columbia  853.
 9 Florida             2786.
10 Georgia             5625.
# ℹ 40 more rows
```

Grouping

Perform Operations By Groups: dplyr

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

```
# Regular data
yts
```

```
# A tibble: 9,794 × 31
    YEAR LocationAbbr LocationDesc TopicType    TopicDesc MeasureDesc DataSource
   <dbl> <chr>        <chr>        <chr>        <chr>     <chr>       <chr>
 1  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 2  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 3  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 4  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 5  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 6  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 7  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 8  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 9  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
10  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
# ℹ 9,784 more rows
# ℹ 24 more variables: Response <chr>, Data_Value_Unit <chr>,
#   Data_Value_Type <chr>, Data_Value <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Data_Value_Std_Err <dbl>,
#   Low_Confidence_Limit <dbl>, High_Confidence_Limit <dbl>, Sample_Size <dbl>,
#   Gender <chr>, Race <chr>, Age <chr>, Education <chr>, GeoLocation <chr>,
#   TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, …
```

Perform Operations By Groups: dplyr

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

```
yts_grouped <- yts %>% group_by(Response)
yts_grouped
```

```
# A tibble: 9,794 × 31
# Groups:   Response [4]
    YEAR LocationAbbr LocationDesc TopicType    TopicDesc MeasureDesc DataSource
   <dbl> <chr>        <chr>        <chr>        <chr>     <chr>       <chr>
 1  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 2  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 3  2015 AZ           Arizona      Tobacco Use… Cessatio… Percent of… YTS
 4  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 5  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 6  2015 AZ           Arizona      Tobacco Use… Cessatio… Quit Attem… YTS
 7  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 8  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
 9  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
10  2015 AZ           Arizona      Tobacco Use… Cigarett… Smoking St… YTS
# ℹ 9,784 more rows
# ℹ 24 more variables: Response <chr>, Data_Value_Unit <chr>,
#   Data_Value_Type <chr>, Data_Value <dbl>, Data_Value_Footnote_Symbol <chr>,
#   Data_Value_Footnote <chr>, Data_Value_Std_Err <dbl>,
#   Low_Confidence_Limit <dbl>, High_Confidence_Limit <dbl>, Sample_Size <dbl>,
#   Gender <chr>, Race <chr>, Age <chr>, Education <chr>, GeoLocation <chr>,
#   TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, …
```

Summarize the data: `dplyr summarize()` function

`summarize` is a helpful function to use after `group_by()`. It creates a summary table of a column you're interested in.

```
yts %>%
  summarize(mean_value = mean(Data_Value, na.rm = TRUE))
```

```
# A tibble: 1 × 1
  mean_value
       <dbl>
1       21.0
```

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 `Data_Value` (percent of respondents) by group:

```
yts_grouped %>%
  summarize(mean_value = mean(Data_Value, na.rm = TRUE))
```

```
# A tibble: 4 × 2
  Response mean_value
  <chr>         <dbl>
1 Current        9.68
2 Ever          26.1
3 Frequent       3.48
4 <NA>          53.5
```

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

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

```
yts %>%
  group_by(Response) %>%
  summarize(mean_value = mean(Data_Value, na.rm = TRUE),
            max_value = max(Data_Value, na.rm = TRUE))
```

```
# A tibble: 4 × 3
  Response mean_value max_value
  <chr>         <dbl>     <dbl>
1 Current        9.68      40.6
2 Ever          26.1       98
3 Frequent       3.48      23.9
4 <NA>          53.5       81.9
```

`group_by` with `mutate` - Useful for comparisons

Use `group_by` to calculate the mean value for each year. We can use `mutate` to add it as a column.

```
yts_year <- yts %>%
  group_by(YEAR) %>%
  mutate(year_avg = mean(Data_Value, na.rm = TRUE)) %>%
  select(LocationDesc, Data_Value, year_avg)
```

`group_by` with `mutate` - Useful for comparisons

Create a "difference" variable:

```
yts_year %>% mutate(Diff = Data_Value - year_avg)
```

```
# A tibble: 9,794 × 5
# Groups:   YEAR [17]
    YEAR LocationDesc Data_Value year_avg   Diff
   <dbl> <chr>             <dbl>    <dbl>  <dbl>
 1  2015 Arizona            NA       15.2  NA
 2  2015 Arizona            NA       15.2  NA
 3  2015 Arizona            NA       15.2  NA
 4  2015 Arizona            NA       15.2  NA
 5  2015 Arizona            NA       15.2  NA
 6  2015 Arizona            NA       15.2  NA
 7  2015 Arizona             3.2     15.2 -12.0
 8  2015 Arizona             3.2     15.2 -12.0
 9  2015 Arizona             3.1     15.2 -12.1
10  2015 Arizona            12.5     15.2  -2.66
# ℹ 9,784 more rows
```

Use `n()` for sample size by group

There are other functions, such as `n()` count the number of observations.

```
yts %>%
  group_by(YEAR) %>%
  summarize(n = n(),
            mean = mean(Data_Value, na.rm = TRUE))
```

```
# A tibble: 17 × 3
    YEAR     n  mean
   <dbl> <int> <dbl>
 1  1999   372  26.1
 2  2000  1224  26.7
 3  2001   426  23.4
 4  2002  1016  25.2
 5  2003   498  21.3
 6  2004   611  20.7
 7  2005   636  21.8
 8  2006   518  21.8
 9  2007   516  20.0
10  2008   483  18.2
11  2009   686  18.3
12  2010   447  17.8
13  2011   521  17.8
14  2012   244  15.5
15  2013   685  16.7
16  2014   334  15.7
17  2015   577  15.2
```

Iterative summaries

Iterative summaries: `dplyr` `summarize()` and `across()` functions

Use the `across` function with `summarize()` to summarize across multiple columns of your data.

```
yts %>%
  group_by(YEAR) %>%
  summarize(across(
    c(Data_Value, Data_Value_Std_Err, Sample_Size),
    ~ mean(.x, na.rm = TRUE)
  ))
```

```
# A tibble: 17 × 4
    YEAR Data_Value Data_Value_Std_Err Sample_Size
   <dbl>      <dbl>               <dbl>       <dbl>
 1  1999       26.1                1.98       1591.
 2  2000       26.7                2.03       1743.
 3  2001       23.4                1.79       2060.
 4  2002       25.2                1.81       2653.
 5  2003       21.3                1.92       2325.
 6  2004       20.7                1.84       1246.
 7  2005       21.8                2.17       1017.
 8  2006       21.8                2.15       1191.
 9  2007       20.0                1.96       1093.
10  2008       18.2                1.73       1203.
11  2009       18.3                1.90       1033.
12  2010       17.8                1.71       1202.
13  2011       17.8                1.84       1274.
14  2012       15.5                1.58       1053.
15  2013       16.7                1.75       1158.
```

Select different columns based on the data class

select helpers (`??tidyr_tidy_select`) are great. We could also use:

- is.numeric
- is.character
- is.factor

to look for data.

Iterative summaries: `dplyr` `summarize()` and `across()` functions

Select only numeric columns.

```
is.numeric(1)
```

```
[1] TRUE
```

```
yts %>%
  summarize(across( where(is.numeric), ~ mean(.x, na.rm = TRUE)))
```

```
# A tibble: 1 × 7
   YEAR Data_Value Data_Value_Std_Err Low_Confidence_Limit High_Confidence_Limit
  <dbl>      <dbl>              <dbl>                <dbl>                 <dbl>
1 2006.       21.0               1.87                 17.3                  24.6
# ℹ 2 more variables: Sample_Size <dbl>, DisplayOrder <dbl>
```

Changing the order of character data - factors

```
yts %>% count(Education)
```

```
# A tibble: 2 × 2
  Education         n
  <chr>         <int>
1 High School    4588
2 Middle School  5206
```

```
yts %>%
  mutate(Education = factor(
    Education,
    levels = c("Middle School", "High School")
  )) %>%
  count(Education)
```

```
# A tibble: 2 × 2
  Education         n
  <fct>         <int>
1 Middle School  5206
2 High School    4588
```

`summary()` Function

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

```
summary(yts)
```

```
     YEAR      LocationAbbr       LocationDesc        TopicType
 Min.   :1999   Length:9794        Length:9794        Length:9794
 1st Qu.:2002   Class :character   Class :character   Class :character
 Median :2006   Mode  :character   Mode  :character   Mode  :character
 Mean   :2006
 3rd Qu.:2010
 Max.   :2015

  TopicDesc         MeasureDesc         DataSource          Response
 Length:9794        Length:9794        Length:9794        Length:9794
 Class :character   Class :character   Class :character   Class :character
 Mode  :character   Mode  :character   Mode  :character   Mode  :character



 Data_Value_Unit    Data_Value_Type      Data_Value
 Length:9794        Length:9794        Min.   : 0.00
 Class :character   Class :character   1st Qu.: 3.20
 Mode  :character   Mode  :character   Median :11.30
                                       Mean   :20.97
                                       3rd Qu.:39.10
                                       Max.   :98.00
```

Summary

- summary stats (`mean()`) work with `pull()`
- `count(x)`: what unique values do you have?
- `group_by()`: changes all subsequent functions
  - combine with `summarize()` to get statistics per group
  - combine with `across()` to programmatically select columns
- `summary(x)`: quantile information

https://sisbid.github.io/Data-Wrangling/labs/data-summarization-lab.Rmd