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**:

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

```
[1] 4.5
```

```r
mean(x, na.rm = TRUE) # Remove NAs if needed
```

```
[1] 4.5
```

Statistical summarization

Summarization on a `data.frame/tibble`:

```
mtcars %>% pull(hp) %>% mean() # alt: pull(mtcars, hp) %>% mean()
```

```
[1] 146.6875
```

```
mean(mtcars$hp)
```

```
[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 Response Data_Value_Unit
  <dbl> <chr>        <chr>        <chr>              <chr>     <chr>       <chr>      <chr>    <chr>
1  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Percent of… YTS        <NA>     %
2  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Percent of… YTS        <NA>     %
3  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Percent of… YTS        <NA>     %
4  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Quit Attem… YTS        <NA>     %
5  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Quit Attem… YTS        <NA>     %
6  2015 AZ           Arizona      Tobacco Use – Sur… Cessatio… Quit Attem… YTS        <NA>     %
# ℹ 22 more variables: 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>, StratificationID4 <chr>, SubMeasureID <chr>,
#   DisplayOrder <dbl>
```

Column to vector

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

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

```
   [1] "Arizona"                   "Arizona"                   "Arizona"
   [4] "Arizona"                   "Arizona"                   "Arizona"
   [7] "Arizona"                   "Arizona"                   "Arizona"
  [10] "Arizona"                   "Arizona"                   "Arizona"
  [13] "Arizona"                   "Arizona"                   "Arizona"
  [16] "Arizona"                   "Arizona"                   "Arizona"
  [19] "Arizona"                   "Arizona"                   "Arizona"
  [22] "Arizona"                   "Arizona"                   "Arizona"
  [25] "Connecticut"               "Connecticut"               "Connecticut"
  [28] "Connecticut"               "Connecticut"               "Connecticut"
  [31] "Connecticut"               "Connecticut"               "Connecticut"
  [34] "Connecticut"               "Connecticut"               "Connecticut"
  [37] "Connecticut"               "Connecticut"               "Connecticut"
  [40] "Connecticut"               "Connecticut"               "Connecticut"
  [43] "Connecticut"               "Connecticut"               "Connecticut"
  [46] "Connecticut"               "Connecticut"               "Connecticut"
  [49] "Connecticut"               "Connecticut"               "Connecticut"
  [52] "Connecticut"               "Connecticut"               "Connecticut"
  [55] "Connecticut"               "Connecticut"               "Connecticut"
  [58] "Connecticut"               "Connecticut"               "Connecticut"
  [61] "Connecticut"               "Connecticut"               "Connecticut"
  [64] "Connecticut"               "Connecticut"               "Connecticut"
  [67] "Connecticut"               "Connecticut"               "Connecticut"
```

Length and unique

`unique(x)` will return the unique elements of `x`

```
unique(locations)
```

```
 [1] "Arizona"                    "Connecticut"                "Georgia"                   
 [4] "Hawaii"                     "Illinois"                   "Louisiana"                 
 [7] "Mississippi"                "Utah"                       "Missouri"                  
[10] "National (States and DC)"   "Nebraska"                   "New Jersey"                
[13] "North Carolina"             "North Dakota"               "Pennsylvania"              
[16] "South Carolina"             "West Virginia"              "Alabama"                   
[19] "Delaware"                   "Minnesota"                  "Guam"                      
[22] "Ohio"                       "Indiana"                    "Kansas"                    
[25] "Oklahoma"                   "Wisconsin"                  "Michigan"                  
[28] "New Hampshire"              "Arkansas"                   "Kentucky"                  
[31] "Iowa"                       "South Dakota"               "Virginia"                  
[34] "Puerto Rico"                "Rhode Island"               "New Mexico"                
[37] "Tennessee"                  "Vermont"                    "Virgin Islands"            
[40] "California"                 "Idaho"                      "Florida"                   
[43] "Maryland"                   "Massachusetts"              "New York"                  
[46] "Maine"                      "Colorado"                   "District of Columbia"      
[49] "Texas"                      "Wyoming"                   
```

Length and unique

`length` will tell you the length of a vector. Combined with `unique`, tells you the number of unique elements:

```
length(unique(locations))
```

```
[1] 50
```

Counting 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.

```
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 Response
   <dbl> <chr>        <chr>        <chr>             <chr>     <chr>       <chr>      <chr>
 1  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 2  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 3  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 4  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 5  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 6  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 7  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
 8  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
 9  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
10  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Ever
# ℹ 9,784 more rows
# ℹ 22 more variables: 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>, Educat
#   GeoLocation <chr>, TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, StratificationID1 <ch
#   StratificationID2 <chr>, StratificationID3 <chr>, StratificationID4 <chr>, SubMeasureID <ch
#   DisplayOrder <dbl>
```

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 Response
   <dbl> <chr>        <chr>        <chr>             <chr>     <chr>       <chr>      <chr>
 1  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 2  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 3  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Percent of… YTS        <NA>
 4  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 5  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 6  2015 AZ           Arizona      Tobacco Use – Su… Cessatio… Quit Attem… YTS        <NA>
 7  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
 8  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
 9  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Current
10  2015 AZ           Arizona      Tobacco Use – Su… Cigarett… Smoking St… YTS        Ever
# ℹ 9,784 more rows
# ℹ 22 more variables: 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>, Educat
#   GeoLocation <chr>, TopicTypeId <chr>, TopicId <chr>, MeasureId <chr>, StratificationID1 <ch
#   StratificationID2 <chr>, StratificationID3 <chr>, StratificationID4 <chr>, SubMeasureID <ch
#   DisplayOrder <dbl>
```

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`:

```r
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.

```r
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.

```
# General format - Not the code!
across({ columns to go across }, ~ { summarization_function(.x, na.rm = ..) })
```

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

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

```
# General format - Not the code!
across({ columns to go across }, ~ { summarization_function(.x, na.rm = ..) })
```

```
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.
```

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

Another example using select helpers (`??tidyr_tidy_select`):

```
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 Sample_Size Di
  <dbl>      <dbl>              <dbl>                <dbl>                 <dbl>       <dbl>
1 2006.       21.0               1.87                 17.3                  24.6       1505.
```

Data Summarization on data frames

- Basic statistical summarization for numeric data
  - `rowMeans(x)`: takes the means of each row of x
  - `colMeans(x)`: takes the means of each column of x
  - `rowSums(x)`: takes the sum of each row of x
  - `colSums(x)`: takes the sum of each column of x
  - `summary(x)`: for data frames, displays the quantile information

`summary()` Function

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

```
summary(yts)
```

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

 MeasureDesc         DataSource          Response         Data_Value_Unit    Data_Value_Type
 Length:9794        Length:9794        Length:9794        Length:9794        Length:9794
 Class  :character  Class  :character  Class  :character  Class  :character  Class  :character
 Mode   :character  Mode   :character  Mode   :character  Mode   :character  Mode   :character



   Data_Value    Data_Value_Footnote_Symbol Data_Value_Footnote Data_Value_Std_Err Low_Confiden
 Min.   : 0.00   Length:9794                Length:9794         Min.   : 0.000     Min.   : 0.0
 1st Qu.: 3.20   Class  :character          Class  :character   1st Qu.: 0.600     1st Qu.: 1.9
 Median :11.30   Mode   :character          Mode   :character   Median : 1.300     Median : 8.5
 Mean   :20.97                                                  Mean   : 1.874     Mean   :17.3
 3rd Qu.:39.10                                                  3rd Qu.: 2.500     3rd Qu.:31.6
 Max.   :98.00                                                  Max.   :16.100     Max.   :97.
```

Summary

- summary stats (`mean()`) work with `pull()`
- `count(x)`: what unique values do you have?
  - `pull()` to get vectors
  - `unique()` combined with `length()`
- `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