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`:

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

```
[1] 146.6875
```

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

```
[1] 146.6875
```

Youth Tobacco Survey

Here we will be using the Youth Tobacco Survey data:

```r
yts <-
  read_csv("http://jhudatascience.org/intro_to_r/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       
# … with 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()`.

```r
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"
```

Length and unique

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

```
unique(locations)
```

```
 [1] "Arizona"                    "Connecticut"
 [3] "Georgia"                    "Hawaii"
 [5] "Illinois"                   "Louisiana"
 [7] "Mississippi"                "Utah"
 [9] "Missouri"                   "National (States and DC)"
[11] "Nebraska"                   "New Jersey"
[13] "North Carolina"             "North Dakota"
[15] "Pennsylvania"               "South Carolina"
[17] "West Virginia"              "Alabama"
[19] "Delaware"                   "Minnesota"
[21] "Guam"                       "Ohio"
[23] "Indiana"                    "Kansas"
[25] "Oklahoma"                   "Wisconsin"
[27] "Michigan"                   "New Hampshire"
[29] "Arkansas"                   "Kentucky"
[31] "Iowa"                       "South Dakota"
[33] "Virginia"                   "Puerto Rico"
[35] "Rhode Island"               "New Mexico"
[37] "Tennessee"                  "Vermont"
[39] "Virgin Islands"             "California"
[41] "Idaho"                      "Florida"
[43] "Maryland"                   "Massachusetts"
[45] "New York"                   "Maine"
```

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
```

`table` and `dplyr`: `count`

`table(x)` will return a frequency table of unique elements of `x`

```
table(locations)
```

```
locations
                 Alabama                  Arizona                 Arkansas 
                     378                      240                      210 
              California                 Colorado              Connecticut 
                      96                       48                      384 
                Delaware     District of Columbia                  Florida 
                     312                       48                       96 
                 Georgia                     Guam                   Hawaii 
                     282                       48                      270 
                   Idaho                 Illinois                  Indiana 
                      48                      282                      264 
                    Iowa                   Kansas                 Kentucky 
                     276                      186                      255 
               Louisiana                    Maine                 Maryland 
                     240                       48                       96 
           Massachusetts                 Michigan                Minnesota 
                      48                      138                      141 
             Mississippi                 Missouri National (States and DC) 
                     567                      294                       26 
                Nebraska            New Hampshire               New Jersey 
                     234                      180                      387 
              New Mexico                 New York           North Carolina 
                      24                       90                      366 
```

table and dplyr: count

Use `count` directly on a data.frame and column without needing to use `pull()`.

```
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
# … with 40 more rows
```

table and 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
# … with 136 more rows
```

table and 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
# … with 136 more rows
```

`table` and `dplyr`: `count`

Note: `count()` includes NAs but `table()` does not

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       
# … with 9,784 more rows, and 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>, …
```

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       
# … with 9,784 more rows, and 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>, …
```

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.

```r
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 - just add data

We can also use `mutate` to calculate the mean value for each year and add it as a column:

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

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

Counting

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 = ..) })
```

```
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.
```

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

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

```
yts %>%
  summarize(across( starts_with("Data"), ~ range(.x, na.rm = TRUE)))
```

```
# A tibble: 2 × 7
  DataSource Data_Value_Unit Data_Value_Type Data_Value Data_Value_Footnote_Sym…
  <chr>      <chr>           <chr>                <dbl> <chr>
1 NYTS       %               Percentage               0 *
2 YTS        %               Percentage              98 *
# … with 2 more variables: Data_Value_Footnote <chr>, Data_Value_Std_Err <dbl>
```

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        
 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  
```

Summary

- summary stats (`mean()`) work with `pull()`
- `count(x)`: what unique values do you have?
    - `pull()` + `table()`
    - `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