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
```

GUT CHECK!

What kind of object do we need to run summary operators like `mean()` ?

A. A vector of numbers

B. A vector of characters

C. A dataset

UFO Data

Let's use the UFO data again:

```
ufo <- read_csv("https://sisbid.github.io/Data-Wrangling/data/ufo/ufo_data_complete.csv")
ufo <- ufo %>% rename(duration_s = `duration (seconds)`)
head(ufo)
```

```
# A tibble: 6 × 11
  datetime  city  state country shape duration_s `duration (hours/min)` comments
  <chr>     <chr> <chr> <chr>   <chr>      <dbl> <chr>                  <chr>
1 10/10/19… san … tx    us      cyli…       2700 45 minutes             This ev…
2 10/10/19… lack… tx    <NA>    light       7200 1-2 hrs                1949 La…
3 10/10/19… ches… <NA>  gb      circ…         20 20 seconds             Green/O…
4 10/10/19… edna  tx    us      circ…         20 1/2 hour               My olde…
5 10/10/19… kane… hi    us      light        900 15 minutes             AS a Ma…
6 10/10/19… bris… tn    us      sphe…        300 5 minutes              My fath…
# ℹ 3 more variables: `date posted` <chr>, latitude <chr>, longitude <chr>
```

Column to vector

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

```
ufo_shapes <- ufo %>% pull(shape)
ufo_shapes
```

```
  [1] "cylinder"  "light"     "circle"    "circle"    "light"     "sphere"
  [7] "circle"    "disk"      "disk"      "disk"      "circle"    "fireball"
 [13] "disk"      "unknown"   "oval"      "circle"    "disk"      "disk"
 [19] "light"     "light"     "other"     "disk"      "light"     "light"
 [25] "oval"      "cigar"     "other"     "light"     "rectangle" "chevron"
 [31] "triangle"  "oval"      "unknown"   "sphere"    "unknown"   "light"
 [37] "disk"      "circle"    "other"     "formation" "triangle"  "chevron"
 [43] "disk"      "triangle"  "unknown"   "light"     "unknown"   "disk"
 [49] "triangle"  "triangle"  "unknown"   "triangle"  "unknown"   "triangle"
 [55] "formation" "unknown"   "cigar"     "unknown"   "sphere"    "light"
 [61] "other"     "cigar"     "rectangle" "light"     "sphere"    NA
 [67] NA          NA          "triangle"  "light"     "light"     "cylinder"
 [73] "delta"     "sphere"    "triangle"  "other"     "disk"      "changing"
 [79] "fireball"  "cylinder"  "cigar"     "circle"    "triangle"  "triangle"
 [85] "light"     "egg"       "fireball"  "changing"  "egg"       "unknown"
 [91] "other"     "fireball"  "sphere"    "circle"    "light"     "light"
 [97] "sphere"    "flash"     "triangle"  "chevron"   "oval"      "oval"
[103] "flash"     "other"     "cylinder"  "triangle"  "oval"      "light"
[109] "light"     "unknown"   "diamond"   "triangle"  "formation" "other"
[115] "circle"    "rectangle" "changing"  "triangle"  "triangle"  "flash"
[121] "unknown"   "circle"    "sphere"    "unknown"   "other"     "circle"
[127] "egg"       "cross"     "light"     "oval"      "sphere"    "other"
[133] "light"     "oval"      "formation" "light"     "unknown"   "sphere"
```

Check for NAs

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

```
sum(is.na(ufo_shapes))
```

```
[1] 3118
```

dplyr: count

Use `count` directly on a data.frame and column: count the number of rows in each group.

```
ufo %>% count(shape)
```

```
# A tibble: 30 × 2
   shape        n
   <chr>    <int>
 1 changed      1
 2 changing  2140
 3 chevron   1007
 4 cigar     2241
 5 circle    8453
 6 cone       367
 7 crescent     2
 8 cross      265
 9 cylinder  1382
10 delta        8
# ℹ 20 more rows
```

dplyr: count

Multiple columns listed further subdivides the count.

```
ufo %>% count(shape, country)
```

```
# A tibble: 139 × 3
   shape    country     n
   <chr>    <chr>   <int>
 1 changed  us          1
 2 changing au         10
 3 changing ca         73
 4 changing de          2
 5 changing gb         47
 6 changing us       1708
 7 changing <NA>      300
 8 chevron  au          3
 9 chevron  ca         38
10 chevron  de          1
# ℹ 129 more rows
```

dplyr: count

Option to sort the results with `sort = TRUE`

```
ufo %>% count(shape, sort = TRUE)
```

```
# A tibble: 30 × 2
   shape        n
   <chr>    <int>
 1 light    17872
 2 triangle  8489
 3 circle    8453
 4 fireball  6562
 5 unknown   6319
 6 other     6247
 7 disk      6005
 8 sphere    5755
 9 oval      4119
10 <NA>      3118
# ℹ 20 more rows
```

GUT CHECK!

The `count()` function can help us tally:

A. Sample size

B. Rows per each category

C. How many categories

dplyr: count

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

```
# Add up "duration_s" for each "shape" category
ufo %>% count(shape, wt = duration_s)
```

```
# A tibble: 30 × 2
   shape            n
   <chr>        <dbl>
 1 changed       3600
 2 changing   4137860.
 3 chevron     492047.
 4 cigar      4045685.
 5 circle    36407947.
 6 cone      26182820
 7 crescent     37810
 8 cross       174465
 9 cylinder   4400398.
10 delta        16155
# ℹ 20 more rows
```

Grouping

Perform Operations By Groups: dplyr

Regular data…

```
# No group_by()
ufo
```

```
# A tibble: 88,875 × 11
   datetime city  state country shape duration_s `duration (hours/min)` comments
   <chr>    <chr> <chr> <chr>   <chr>      <dbl> <chr>                  <chr>
 1 10/10/1… san … tx    us      cyli…       2700 45 minutes             This ev…
 2 10/10/1… lack… tx    <NA>    light       7200 1-2 hrs                1949 La…
 3 10/10/1… ches… <NA>  gb      circ…         20 20 seconds             Green/O…
 4 10/10/1… edna  tx    us      circ…         20 1/2 hour               My olde…
 5 10/10/1… kane… hi    us      light        900 15 minutes             AS a Ma…
 6 10/10/1… bris… tn    us      sphe…        300 5 minutes              My fath…
 7 10/10/1… pena… <NA>  gb      circ…        180 about 3 mins           penarth…
 8 10/10/1… norw… ct    us      disk        1200 20 minutes             A brigh…
 9 10/10/1… pell… al    us      disk         180 3  minutes             Strobe …
10 10/10/1… live… fl    us      disk         120 several minutes         Saucer …
# ℹ 88,865 more rows
# ℹ 3 more variables: `date posted` <chr>, latitude <chr>, longitude <chr>
```

Perform Operations By Groups: dplyr

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

```
ufo_grouped <- ufo %>% group_by(shape)
ufo_grouped
```

```
# A tibble: 88,875 × 11
# Groups:   shape [30]
   datetime city  state country shape duration_s `duration (hours/min)` comments
   <chr>    <chr> <chr> <chr>   <chr>      <dbl> <chr>                  <chr>
 1 10/10/1… san … tx    us      cyli…       2700 45 minutes             This ev…
 2 10/10/1… lack… tx    <NA>    light       7200 1-2 hrs                1949 La…
 3 10/10/1… ches… <NA>  gb      circ…         20 20 seconds             Green/O…
 4 10/10/1… edna  tx    us      circ…         20 1/2 hour               My olde…
 5 10/10/1… kane… hi    us      light        900 15 minutes             AS a Ma…
 6 10/10/1… bris… tn    us      sphe…        300 5 minutes              My fath…
 7 10/10/1… pena… <NA>  gb      circ…        180 about 3 mins           penarth…
 8 10/10/1… norw… ct    us      disk        1200 20 minutes             A brigh…
 9 10/10/1… pell… al    us      disk         180 3  minutes             Strobe …
10 10/10/1… live… fl    us      disk         120 several minutes         Saucer …
# ℹ 88,865 more rows
# ℹ 3 more variables: `date posted` <chr>, latitude <chr>, longitude <chr>
```

Summarize the grouped data

We can summarize grouped data using `summarize()`:

```
ufo_grouped %>%
  summarize(mean_value = mean(duration_s, na.rm = TRUE))
```

```
# A tibble: 30 × 2
   shape    mean_value
   <chr>         <dbl>
 1 changed       3600 
 2 changing      1934.
 3 chevron        489.
 4 cigar         1805.
 5 circle        4308.
 6 cone         71343.
 7 crescent     18905 
 8 cross          658.
 9 cylinder      3184.
10 delta         2019.
# ℹ 20 more rows
```

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

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

```
ufo %>%
  group_by(shape) %>%
  summarize(mean_value = mean(duration_s, na.rm = TRUE),
            max_value = max(duration_s, na.rm = TRUE))
```

```
# A tibble: 30 × 3
   shape    mean_value max_value
   <chr>         <dbl>     <dbl>
 1 changed       3600       3600
 2 changing      1934.    172800
 3 chevron        489.     43200
 4 cigar         1805.   1814400
 5 circle        4308.  10526400
 6 cone         71343.  25248000
 7 crescent     18905      37800
 8 cross          658.     10800
 9 cylinder      3184.   2631600
10 delta         2019.     14400
# ℹ 20 more rows
```

`group_by` with `mutate` - Useful for comparisons

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

```
ufo_compare <- ufo %>%
  group_by(shape) %>%
  mutate(duration_shape_avg = mean(duration_s, na.rm = TRUE)) %>%
  select(shape, duration_s, duration_shape_avg)
ufo_compare
```

```
# A tibble: 88,875 × 3
# Groups:   shape [30]
   shape    duration_s duration_shape_avg
   <chr>         <dbl>              <dbl>
 1 cylinder       2700              3184.
 2 light          7200             12256.
 3 circle           20              4308.
 4 circle           20              4308.
 5 light           900             12256.
 6 sphere          300             20411.
 7 circle          180              4308.
 8 disk           1200              2018.
 9 disk            180              2018.
10 disk            120              2018.
# ℹ 88,865 more rows
```

`group_by` with `mutate` - Useful for comparisons

Create a “difference” variable:

```
ufo_compare %>% mutate(diff = duration_s - duration_shape_avg)
```

```
# A tibble: 88,875 × 4
# Groups:   shape [30]
   shape    duration_s duration_shape_avg    diff
   <chr>         <dbl>              <dbl>   <dbl>
 1 cylinder       2700              3184.   -484.
 2 light          7200             12256.  -5056.
 3 circle           20              4308.  -4288.
 4 circle           20              4308.  -4288.
 5 light           900             12256. -11356.
 6 sphere          300             20411. -20111.
 7 circle          180              4308.  -4128.
 8 disk           1200              2018.   -818.
 9 disk            180              2018.  -1838.
10 disk            120              2018.  -1898.
# ℹ 88,865 more rows
```

Iterative summaries

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

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

```
dropouts <- read_delim("https://sisbid.github.io/Data-Wrangling/data/dropouts.txt", delim = "/")
dropouts %>%
  group_by(ETHNIC) %>%
  summarize(across(c(D9, D10, D11, D12), ~sum(.x)))
```

```
# A tibble: 9 × 5
  ETHNIC    D9   D10   D11   D12
   <dbl> <dbl> <dbl> <dbl> <dbl>
1      0  1226   299   308   478
2      1   103    82   124   265
3      2   118   114   132   767
4      3    37    25    34   135
5      4    34    30    51   252
6      5  4433  3543  4723 16220
7      6   817   617   875  2890
8      7   768   878  1432  3998
9      9   162   155   194   636
```

Select different columns based on the data class

Use `?tidyr_tidy_select` functions like `where(is.numeric)`!

```
dropouts %>%
  group_by(ETHNIC) %>%
  summarize(across( where(is.numeric), ~ sum(.x, na.rm = TRUE)))
```

```
# A tibble: 9 × 18
  ETHNIC     E7     E8     E9    E10    E11    E12   EUS    ETOT    D7    D8
   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <dbl>   <dbl> <dbl> <dbl>
1      0   2922   2709   3140   3291   2776   2345    43   11595    66    52
2      1   2523   2589   2808   2886   2889   3020    33   11636    16    15
3      2  44199  43898  43343  43757  45840  43001   596  176537    69    37
4      3   2390   2284   2436   2352   2471   2571    42    9872     9     7
5      4  12210  12883  13429  14037  14273  14469   233   56441    12     8
6      5 254745 252583 264302 260201 252458 253193  2680 1032834   682   550
7      6  26383  26786  28497  28731  28696  30799   485  117208   217   164
8      7 113498 115139 115593 117592 119308 121771  1829  476093   323   267
9      9  14704  13469  13999  13238  12810  13000   138   53185    59    38
# ℹ 7 more variables: D9 <dbl>, D10 <dbl>, D11 <dbl>, D12 <dbl>, DUS <dbl>,
#   DTOT <dbl>, YEAR <dbl>
```

`summary()` Function

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

```
summary(dropouts)
```

```
   CDS_CODE             ETHNIC        GENDER                E7        
 Length:59599       Min.   :0.000   Length:59599       Min.   :  0.000  
 Class :character   1st Qu.:2.000   Class :character   1st Qu.:  0.000  
 Mode  :character   Median :5.000   Mode  :character   Median :  0.000  
                    Mean   :4.737                      Mean   :  7.946  
                    3rd Qu.:7.000                      3rd Qu.:  3.000  
                    Max.   :9.000                      Max.   :373.000  
       E8                 E9                E10                E11         
 Min.   :  0.000   Min.   :  0.00   Min.   :  0.000   Min.   :  0.000  
 1st Qu.:  0.000   1st Qu.:  0.00   1st Qu.:  0.000   1st Qu.:  0.000  
 Median :  0.000   Median :  0.00   Median :  0.000   Median :  0.000  
 Mean   :  7.925   Mean   :  8.18   Mean   :  8.156   Mean   :  8.079  
 3rd Qu.:  3.000   3rd Qu.:  2.00   3rd Qu.:  2.000   3rd Qu.:  2.000  
 Max.   :363.000   Max.   :567.00   Max.   :537.000   Max.   :620.000  
      E12                EUS               ETOT                D7          
 Min.   :  0.000   Min.   : 0.000   Min.   :   0.00   Min.   : 0.00000  
 1st Qu.:  0.000   1st Qu.: 0.000   1st Qu.:   0.00   1st Qu.: 0.00000  
 Median :  0.000   Median : 0.000   Median :   1.00   Median : 0.00000  
 Mean   :  8.124   Mean   : 0.102   Mean   :  32.64   Mean   : 0.02438  
 3rd Qu.:  2.000   3rd Qu.: 0.000   3rd Qu.:   9.00   3rd Qu.: 0.00000  
 Max.   :490.000   Max.   :77.000   Max.   :2127.00   Max.   :15.00000  
      D8                 D9                D10                D11          
 Min.   :0.00000   Min.   :  0.0000   Min.   : 0.00000   Min.   :  0.0000  
```

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 select several columns
- `summary(x)`: quantile information

https://sisbid.github.io/Data-Wrangling/08_Data_Summarization/lab/data-summarization-lab.Rmd