Numerical Variable Summaries

We now know how to summarize categorical data and we've learned the basics of `ggplot2`. Now we're ready to investigate how to summarize numeric variables. Recall:

- Numeric (Quantitative) variable - entries are a numerical value where math can be performed

As before, our goal is to describe the **distribution** of the variable. We talked about this briefly:

- For a single numeric variable, describe the distribution via
  - Shape: Histogram, Density plot, …
  - Measures of center: Mean, Median, …
  - Measures of spread: Variance, Standard Deviation, Quartiles, IQR, …
- For two numeric variables, describe the distribution via
  - Shape: Scatter plot, …
  - Measures of linear relationship: Covariance, Correlation

First, let's read in the appendicitis data from the previous lecture.

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
✔ dplyr     1.1.4     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.4     ✔ tibble    3.2.1
✔ lubridate 1.9.2     ✔ tidyr     1.3.0
✔ purrr     1.0.1
── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(readxl)
app_data <- read_excel("app_data.xlsx", sheet = 1)
app_data <- app_data |>
  mutate(BMI = as.numeric(BMI),
         US_Number = as.character(US_Number),
         SexF = factor(Sex, levels = c("female", "male"), labels = c("Female
         DiagnosisF = as.factor(Diagnosis),
         SeverityF = as.factor(Severity))
app_data
```

```
# A tibble: 782 × 61
```

```
      Age   BMI  Sex     Height  Weight  Length_of_Stay  Management    Severity
    <dbl> <dbl> <chr>    <dbl>   <dbl>            <dbl>  <chr>         <chr>
 1 12.7    16.9 female    148    37                  3  conservative
uncomplicated
 2 14.1    31.9 male      147    69.5                2  conservative
uncomplicated
 3 14.1    23.3 female    163    62                  4  conservative
uncomplicated
 4 16.4    20.6 female    165    56                  3  conservative
uncomplicated
 5 11.1    16.9 female    163    45                  3  conservative
uncomplicated
 6 11.0    30.7 male      121    45                  3  conservative
uncomplicated
 7  8.98   19.4 female    140    38.5                3  conservative
uncomplicated
 8  7.06   NA   female     NA    21.5                2  conservative
uncomplicated
 9  7.9    15.7 male      131    26.7                3  conservative
uncomplicated
10 14.3    14.9 male      174    45.5                3  conservative
uncomplicated
# ℹ 772 more rows
# ℹ 53 more variables: Diagnosis_Presumptive <chr>, Diagnosis <chr>,
#   Alvarado_Score <dbl>, Paedriatic_Appendicitis_Score <dbl>,
#   Appendix_on_US <chr>, Appendix_Diameter <dbl>, Migratory_Pain <chr>,
#   Lower_Right_Abd_Pain <chr>, Contralateral_Rebound_Tenderness <chr>,
#   Coughing_Pain <chr>, Nausea <chr>, Loss_of_Appetite <chr>,
#   Body_Temperature <dbl>, WBC_Count <dbl>, Neutrophil_Percentage <dbl>, …
```

Let’s dig in!

Numerical Summaries

We’ll utilize the `summarize()` function along with `group_by()` to find most of our numerical summaries.

As we discussed, we can’t really describe the entire distribution with a single number so we try to summarize different aspects of the distribution. In particular, center and spread.

Measures of Center

We can find the mean and median via the `mean()` and `median()` function.

```
app_data |>
  summarize(mean_BMI = mean(BMI, na.rm = TRUE), med_BMI = median(BMI, na.rm
```

```
# A tibble: 1 × 2
  mean_BMI med_BMI
     <dbl>   <dbl>
1     18.9    18.1
```

We can try to get fancy and do it for all numeric columns. Recall we did this earlier with `across()` and `where(is.numeric)`:

```
app_data |>
  summarize(across(where(is.numeric),
                   list("mean" = mean, "median" = median),
                   .names = "{.fn}_{.col}"))
```

```
# A tibble: 1 × 34
  mean_Age median_Age mean_BMI median_BMI mean_Height median_Height
mean_Weight
     <dbl>      <dbl>    <dbl>      <dbl>       <dbl>         <dbl>
<dbl>
1       NA         NA       NA         NA          NA            NA
NA
# ℹ 27 more variables: median_Weight <dbl>, mean_Length_of_Stay <dbl>,
#   median_Length_of_Stay <dbl>, mean_Alvarado_Score <dbl>,
#   median_Alvarado_Score <dbl>, mean_Paedriatic_Appendicitis_Score <dbl>,
#   median_Paedriatic_Appendicitis_Score <dbl>, mean_Appendix_Diameter
<dbl>,
#   median_Appendix_Diameter <dbl>, mean_Body_Temperature <dbl>,
#   median_Body_Temperature <dbl>, mean_WBC_Count <dbl>,
#   median_WBC_Count <dbl>, mean_Neutrophil_Percentage <dbl>, …
```

Oh, darn. That's right, we have missing values. We can remove those just for a particular column instead of removing all the rows (as we did with `drop_na()`). This is a bit more complicated but we can specify some additional arguments of the mean and median function in our named list.

```
app_data |>
  summarize(across(where(is.numeric),
                   list("mean" = ~ mean(.x, na.rm = TRUE), "median" = ~ med
                   .names = "{.fn}_{.col}"))
```

```
# A tibble: 1 × 34
  mean_Age median_Age mean_BMI median_BMI mean_Height median_Height
mean_Weight
     <dbl>      <dbl>    <dbl>      <dbl>       <dbl>         <dbl>
<dbl>
1     11.3       11.4     18.9       18.1        148.          150.
43.2
# ℹ 27 more variables: median_Weight <dbl>, mean_Length_of_Stay <dbl>,
#   median_Length_of_Stay <dbl>, mean_Alvarado_Score <dbl>,
#   median_Alvarado_Score <dbl>, mean_Paedriatic_Appendicitis_Score <dbl>,
#   median_Paedriatic_Appendicitis_Score <dbl>, mean_Appendix_Diameter
<dbl>,
#   median_Appendix_Diameter <dbl>, mean_Body_Temperature <dbl>,
#   median_Body_Temperature <dbl>, mean_WBC_Count <dbl>,
#   median_WBC_Count <dbl>, mean_Neutrophil_Percentage <dbl>, …
```

The `~` is a quick way to write a *lambda* or *anonymous* function. Essentially, we are inline doing something like this

```
my_fun <- function(x) {mean(x, na.rm = TRUE)}
```

But a *lambda* function is a shorthand for this where we don't need to give the function a name (since we aren't planning on using it again anyway).

Of course we want these kinds of statistics across groups so we can compare them. We saw how to do this with `group_by()`

```
app_data |>
  group_by(Diagnosis, Sex) |>
  drop_na(Diagnosis, Sex) |>
  summarize(mean_BMI = mean(BMI, na.rm = TRUE), med_BMI = median(BMI, na.rm
```

```
`summarise()` has grouped output by 'Diagnosis'. You can override using the
`.groups` argument.

# A tibble: 4 × 4
# Groups:   Diagnosis [2]
  Diagnosis       Sex    mean_BMI med_BMI
  <chr>           <chr>     <dbl>   <dbl>
1 appendicitis    female     18.6    17.8
2 appendicitis    male       18.3    17.5
3 no appendicitis female     20.2    20.0
4 no appendicitis male       18.7    17.2
```

We can do this similar thing with the fancier version too!

```
app_data |>
  group_by(Diagnosis, Sex) |>
  drop_na(Diagnosis, Sex) |>
  summarize(across(where(is.numeric),
                   list("mean" = ~ mean(.x, na.rm = TRUE), "median" = ~ med
                   .names = "{.fn}_{.col}"))
```

```
`summarise()` has grouped output by 'Diagnosis'. You can override using the
`.groups` argument.

# A tibble: 4 × 36
# Groups:   Diagnosis [2]
  Diagnosis       Sex    mean_Age median_Age mean_BMI median_BMI
mean_Height
  <chr>           <chr>     <dbl>      <dbl>    <dbl>      <dbl>
<dbl>
1 appendicitis    female     11.7       11.9     18.6       17.8
148.
2 appendicitis    male       10.6       11       18.3       17.5
146.
3 no appendicitis female     12.5       12.5     20.2       20.0
152.
4 no appendicitis male       10.7       10.9     18.7       17.2
147.
# ℹ 29 more variables: median_Height <dbl>, mean_Weight <dbl>,
#   median_Weight <dbl>, mean_Length_of_Stay <dbl>,
#   median_Length_of_Stay <dbl>, mean_Alvarado_Score <dbl>,
#   median_Alvarado_Score <dbl>, mean_Paedriatic_Appendicitis_Score <dbl>,
#   median_Paedriatic_Appendicitis_Score <dbl>, mean_Appendix_Diameter
<dbl>,
#   median_Appendix_Diameter <dbl>, mean_Body_Temperature <dbl>,
#   median_Body_Temperature <dbl>, mean_WBC_Count <dbl>, …
```

Great, now we have an easy way to compare the centers of the distribution for each of these numeric variables!

Measures of Spread

Same idea here but we can use the `sd()` and `IQR()` functions.

```
app_data |>
  group_by(Diagnosis, Sex) |>
  drop_na(Diagnosis, Sex) |>
  summarize(sd_BIM = sd(BMI, na.rm = TRUE), IQR_BMI = IQR(BMI, na.rm = TRUE
```

```
`summarise()` has grouped output by 'Diagnosis'. You can override using the
`.groups` argument.

# A tibble: 4 × 4
# Groups:   Diagnosis [2]
  Diagnosis       Sex    sd_BIM IQR_BMI
  <chr>           <chr>   <dbl>   <dbl>
1 appendicitis    female   4.33    4.62
2 appendicitis    male     4.03    5.22
3 no appendicitis female   4.53    5.75
4 no appendicitis male     4.60    5.07
```

Measures of Association Between Two Numeric Variables

We can find the linear associations between two numeric variables with `cor()`.

```
app_data |>
  group_by(Diagnosis, Sex) |>
  drop_na(Diagnosis, Sex) |>
  summarize(correlation = cor(BMI, Age))
```

```
`summarise()` has grouped output by 'Diagnosis'. You can override using the
`.groups` argument.

# A tibble: 4 × 3
# Groups:   Diagnosis [2]
  Diagnosis       Sex    correlation
  <chr>           <chr>        <dbl>
1 appendicitis    female          NA
2 appendicitis    male            NA
3 no appendicitis female          NA
4 no appendicitis male            NA
```

Oh yeah, missing values. Unfortunately, `BaseR` isn't that consistent. To deal with missing values appropriately, we can look at the help.

`use`

> an optional character string giving a method for computing covariances in the presence of missing values. This must be (an abbreviation of) one of the strings “everything”, “all.obs”, “complete.obs”, “na.or.complete”, or “pairwise.complete.obs”.

```r
app_data |>
  group_by(Diagnosis, Sex) |>
  drop_na(Diagnosis, Sex) |>
  summarize(correlation = cor(BMI, Age, use = "pairwise.complete.obs"))
```

```
`summarise()` has grouped output by 'Diagnosis'. You can override using the
`.groups` argument.

# A tibble: 4 × 3
# Groups:   Diagnosis [2]
  Diagnosis       Sex    correlation
  <chr>           <chr>        <dbl>
1 appendicitis    female       0.556
2 appendicitis    male         0.462
3 no appendicitis female       0.413
4 no appendicitis male         0.422
```

Great - we can do all our basic numerical summaries!

Recap!

We tend to describe the center and spread of a numeric variable’s distribution. Often we want to compare across groups and that can be done with `group_by()`.