Notes 02 - Data Summaries

STS 2300 (Spring 2025)

Updated: 2025-01-25

Table of Contents

Reading for Notes 02

There is no specific reading associated with the material in this set of notes. Instead, consider completing a couple of the R Programming modules from the `swirl` package to work on building your foundational R skills. The following would be particularly helpful as you learn more about R:

- Module 1 - Basic Building Blocks
- Module 3 - Sequences and Numbers
- Module 4 - Vectors
- Module 12 - Looking at Data

You can learn more about the `swirl` package here or you can talk to the instructor to get started.

Learning Goals for Notes 02

- Be able to identify proper ways to summarize categorical and quantitative data and to write code to carry this out.
- Be able to conduct summaries by groups when applicable

- Be able to format summaries appropriately for use in future code (and for readability as discussed in Activity 02)
- Be able to use R Markdown to present statistical results that interweave text, code, and output (Activity 02)

Quantitative vs. categorical data and representation in R

In statistics, we commonly split data into two types: **quantitative** (sometimes called numeric) and **categorical** (sometimes called qualitative). You should have learned (or will learn) these definitions of variables in STS 2120.

- **Quantitative data** consists of numbers with numeric meanings. In other words, it would make sense to interpret an average for this data. Quantitative data is usually the result of making counts or measurements.
- **Categorical data** consists of data split into categories. The data may or may not *look* like numbers. If they look like numbers, they have a non-numeric meaning (e.g. zip code).

In R, quantitative data can be stored in several different formats (e.g. int, num, dbl, etc.). We will typically not distinguish between these in our class. We can use the `is.numeric()` function to verify if a scalar, vector, or matrix is being treated as quantitative by R.

Categorical data also may be stored in different formats (e.g. chr, factor). We may discuss differences between these later in the semester. Functions like `is.character()` and `is.factor()` can be used similarly to `is.numeric()`.

If we want to get a quick view of how all the variables in a data frame (or vector) are being stored, we can use the `str()` function.

Practice: Try using `str()` on the `mtcars` data frame from Notes 01. How is R storing each of the variables? Does the results seem to match how we would think of them "statistically"? (Note: You can compare this to the `MA_schools` example using `str()` further down in the notes.)

Answer: All the variables are being stored as numeric (quantitative). At least engine and transmission are really categorical (we could make an argument for cylinders, too).

In some cases, we may need to convert a variable to an appropriate format. Functions like `as.factor()`, `as.numeric()`, etc. can help with this. We will revisit this more as it arises throughout the semester.

Summarizing Quantitative Data

There are many, many ways to summarize numeric data. Below are some common ones that you likely saw (or will see) in STS 2120. The associated R functions are listed next to each.

Measures of data center:

- mean – `mean()` (what we typically think of as the average)
- median – `median()` (the middle number once values are ordered)

Measures of spread:

- standard deviation – `sd()` (a "typical" distance values fall from the mean)
- variance – `var()` (the square of standard deviation. Not often used outside other calculations)
- interquartile range (IQR) – `IQR()` (the 75th percentile minus the 25th percentile. It tells us how spread out the middle 50% of our data is.)

Measures of location:

- minimum – `min()`
- maximum – `max()`
- kth percentile (e.g. 90th) – `quantile(__, probs = .9)`
    - The probs argument is used to choose which percentile we want.
    - A percentile is the value in the data that has _% of other values below it. For example, the 90th percentile has 90% of data values below it.

Other:

- total or sum – `sum()`
- number of observations – `length()`

We can use any of these functions on a *vector of numeric values* to obtain a single number.

Practice: Try this on the `mtcars` data frame to find:

- The mean miles per gallon
- The minimum horsepower
- The 80th percentile for weight
- The standard deviation for displacement

(Hints: Remember you need to use the `$` operator to isolate a vector/column from a data frame. You can use `?` to find the column names for each variable in `mtcars`.)

Answers: See notes02.R.

If we want to create a data frame of our summary with columns for each value (think PROC MEANS in SAS), we can use the `summarize()` function from the `dplyr` package. The `summarize()` function has a data frame as its first argument followed by calculations for as many statistics as we'd like. We can add names to these statistics in our output by preceding the calculation with a name and an equals sign. Below is an example:

```
library(dplyr)
summarize(mtcars,
          min_mpg = min(mpg),
          max_mpg = max(mpg),
          avg_hp = mean(hp),
          sd_hp = sd(hp))
```

```
##   min_mpg max_mpg   avg_hp    sd_hp
## 1    10.4    33.9 146.6875 68.56287
```

In summarize(), the first argument is your data frame. Then, you can include as many other calculations as you want. Each one should have a name for the column, an equals sign, and a calculation.

The resulting output is still a data frame and has named columns that we could reference in future code (if we store the data frame in our environment).

Practice: Try running this code but save the output of the `summarize()` function in an object called `car_sum`. Then reference the average horse power within this new object (Hint: use $ and the name for this column).

Answer: See notes 02.R.

Summarizing Categorical Data

Let's explore the `MA_schools` data frame in the `moderndive` package associated with our textbook. If you haven't installed this package before, you will need to run `install.packages("moderndive")` in your console. Otherwise, you can start with the code below to explore the structure of the data.

```
library(moderndive)
```

```
## Warning: package 'moderndive' was built under R version 4.3.3
```

```
str(MA_schools)
```

```
## spc_tbl_ [332 × 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ school_name     : chr [1:332] "Abington High" "Agawam High" "Amesbury
High" "Andover High" ...
##  $ average_sat_math: num [1:332] 516 514 534 581 592 576 504 505 481 513
...
##  $ perc_disadvan   : num [1:332] 21.5 22.7 14.6 6.3 10.3 10.3 25.6 15.2
23.8 25.5 ...
##  $ size            : Factor w/ 3 levels "small","medium",..: 2 3 3 3 3 3 3
3 1 3 ...
```

Question: Which of these variables are categorical? Which variable(s) would make sense to summarize with a table? (Note: You may also choose to use a function like `head()` to further explore the data)

Answer: school_name and size are both categorical. It would make more sense to use size for a table because school_name is likely unique for each school.

In Activity 01, we saw examples of using the `table()` and `prop.table()` functions as ways of summarizing data with a small number of possible outcomes.

Practice: Create a table of school sizes using `table()`. Then write a second line of code where your code from the first line is inside the `prop.table()` function. What is different about the two outputs?

Answer: The table() function gives us the number of schools in each category (e.g., 28 small schools). The prop.table() function gives us the proportion of schools in each category (e.g., 0.084 = 8.4% of schools are small).

The output from these two functions are **not** data frames. This means they may be difficult to use in future code and calculations.

Summaries by groups

The `summarize()` function has an argument called `.by` that allows us to calculate the same statistic(s) for different groups in our data (e.g., the minimum mpg for cars with automatic transmissions and for cars with manual transmissions). This method produces results as a data frame and is one way to solve our previous issue about the `table()` function producing output that is hard to use in future code and calculations. For example, we can use the `n()` function in `summarize()` along with a `.by = size` argument to calculate the number of schools *by* school size.

```
summarize(MA_schools,
          count = n(),
          .by = size)
```

```
## # A tibble: 3 × 2
##   size   count
##   <fct>  <int>
## 1 medium    69
## 2 large    235
## 3 small     28
```

These are the same results we would get using `table(MA_schools$size)`, but the output is now a data frame (with variables called `size` and `count`) that we can use in future code and calculations. In Notes 03, we will see how we could add a proportion to this table as well.

This same approach can be used to carry out numerical summaries by groups. Let's revisit our `mtcars` summary from before but now summarize by transmission type (`am`).

Practice: See if you can edit your code from before to produce the output below. (Bonus: Add another column called `count` that tells us how many cars have each transmission type)

```
##   am min_mpg max_mpg   avg_hp    sd_hp
## 1  1    15.0    33.9 126.8462 84.06232
## 2  0    10.4    24.4 160.2632 53.90820
```

To summarize by multiple groups, we can put variables inside `c()` for the `.by` argument. For example, suppose I wanted a table of how many cars had certain transmissions and numbers of cylinders.

```
summarize(mtcars,
          count = n(),
          .by = c(am, cyl))
```

```
##   am cyl count
## 1  1   6     3
## 2  1   4     8
## 3  0   6     4
## 4  0   8    12
## 5  0   4     3
## 6  1   8     2
```

I can see that 12 cars have automatic transmission (`am = 0`) and 8 cylinders. We will learn in Activity 02 and Notes 03 some ways to make the output look nicer.

Practice: Update your code from the `mtcars` summary in the practice problem problem above to calculate numerical summaries by transmission and cylinders. Your output should look like this.

```
##   am cyl min_mpg max_mpg    avg_hp    sd_hp
## 1  1   6    19.7    21.0 131.66667 37.52777
## 2  1   4    21.4    33.9  81.87500 22.65542
## 3  0   6    17.8    21.4 115.25000  9.17878
## 4  0   8    10.4    19.2 194.16667 33.35984
## 5  0   4    21.5    24.4  84.66667 19.65536
## 6  1   8    15.0    15.8 299.50000 50.20458
```

Revisiting the Learning Goals for Notes 02

- Be able to identify proper ways to summarize categorical and quantitative data and to write code to carry this out.
    - Use the `mario_kart_auction` data frame from the `moderndive` package to create a table showing counts for two categorical variables.
    - Use the same data frame to summarize the center and spread of a quantitative variable from the data.

- Be able to conduct summaries by groups when applicable
    - Use the `mario_kart_auction` data frame to calculate the median and IQR of the total price for new versus used games.

- Be able to format summaries from raw output to more readable formats. (Activity 02)
    - After completing Activity 02, revisit your median and IQR from the previous learning goal and use the `kableExtra` along with one of the `kable_xxxx()` functions. Change settings so the table does not take up the whole page.

- Be able to use R Markdown to present statistical results that interweave text, code, and output (Activity 02)
    - Use the `kableExtra` package along with R Markdown to create a short document summarizing the `mario_kart_auction` data. Make sure your tables use appropriate themes. Hide your code chunks in the knitted document so that someone who doesn't know R could still read it and understand your summaries.