Today, I am going to cover summarize factors numerically.

The factors only have finite many cases. Therefore, it is possible for us to list all of the potential values with the count. This is called a frequency table.

To test whether a variable is a factor or not in R, we can use the is.factor() function or class() function.

```
is.factor(StudentsPerformance$Gender)
```

```
## [1] FALSE
```

The return value is FALSE, which means that it is not treated as a categorical variable in R yet.

```
class(StudentsPerformance$Gender)
```

```
## [1] "character"
```

The class function returns “character”, which means it is not a factor in R yet.

The R engine doesn’t know that Gender is factor, we need to convert it manually using the function as.factor():

```
#convert Genderto factor
StudentsPerformance$Gender <- as.factor(StudentsPerformance$Gender)
print(is.factor(StudentsPerformance$Gender))
```

```
## [1] TRUE
```

```
print(class(StudentsPerformance$Gender))
```

```
## [1] "factor"
```

Now the gender variable is a factor. To look at the number of cases in a factor, which is called levels, we can use the levels() function:

```
levels(StudentsPerformance$Gender)
```

```
## [1] "female" "male"
```

It shows that gender has two cases, “female” and “male”, in the given data set. To get the frequency table, we can use the table() function in R.

```
table(StudentsPerformance$Gender)
```

```
##
## female   male
##    518    482
```

We can see that there are 518 female test takers and 482 male test takers. We are also interested in finding the **most** frequently occurring case for the factors. We can use the following R command to find the mode of gender variable.

```
names(sort(-table(StudentsPerformance$Gender)))[1]
```

```
## [1] "female"
```

To find the mode of a factor, the R code above performs the following operations:

- Find the frequency table using table () function.
- Sort the frequency table by descending order using sort function with - in front of table
- Return the first variable since it has the largest count

Sometimes, we are interested in the percentage instead of a frequency table. To generate a proportional table, we can use the prop.table() function after we summarize the data using the table() function. For example, we can generate the table of proportions for Gender using the following codes:

```
prop.table(table(StudentsPerformance$Gender))
```

```
##
## female   male
##  0.518  0.482
```

We can also summarize data using a cross table as we did in Microsoft Excel using the pivot table. The cross table is a joint frequency table based on several factors.

To generate a cross table in R, we typically use the xtabs () function with the following two arguments:

- The first parameter starting with ~ specifies the factors based on R formula.
- The second parameter specifies the data set

For example, we can generate a cross table of gender and race. To do it, we first need to make sure that gender and race are factors in R. We can check it using the is.factor() function.

```
is.factor(StudentsPerformance$Gender)
```

```
## [1] TRUE
```

```
is.factor(StudentsPerformance$Race)
```

```
## [1] FALSE
```

We can see that race is not stored as a factor in R memory, we need to convert it to a factor by using the following command

```
StudentsPerformance$Race <- as.factor(StudentsPerformance$Race)
```

Now, both are factors, we can use the xtabs() function:

```
xtabs(~ Gender + Race, data = StudentsPerformance)
```

```
##         Race
## Gender   group A group B group C group D group E
```

```
##   female      36     104     180     129      69
##   male        53      86     139     133      71
```

Note that we only need to provide the variable names in the first parameter and join them using + operator.

If we want to produce proportions instead of count/frequency in a cross table, we need to pass the result of a cross table to prop.table() function and specify a margin parameter:

- margin = 1 means it computes the proportion with respect to the row.
- margin = 2 means it computes the proportion with respect to the column.

For example, to produce a proportion with respect to the row (gender) that is the **first** parameter in the R formula $\sim Gender + Race$, we can use the following R codes:

```
xtab.gender.race <- xtabs(~ Gender + Race, data = StudentsPerformance)
prop.table(xtab.gender.race, margin = 1)
```

```
##         Race
## Gender      group A    group B    group C    group D    group E
##   female 0.06949807 0.20077220 0.34749035 0.24903475 0.13320463
##   male   0.10995851 0.17842324 0.28838174 0.27593361 0.14730290
```

Notice that the sum of each row is 1 since we produce a proportion based on row data.

If you want to produce a proportion with respect to the column (Race) that is the **second** parameter in the R formula $\sim Gender + Race$, we can use the following R codes:

```
xtab.gender.race <- xtabs(~ Gender + Race, data = StudentsPerformance)
prop.table(xtab.gender.race, margin = 2)
```

```
##         Race
## Gender     group A   group B   group C   group D   group E
##   female 0.4044944 0.5473684 0.5642633 0.4923664 0.4928571
##   male   0.5955056 0.4526316 0.4357367 0.5076336 0.5071429
```

Notice that the sum of each column is 1 since we produce a proportion based on column data.

When we have a data frame in R, we can easily summarize the data frame using the summary() function. This function performs the following tasks:

- Continuous variables: return the following five quantities:
  - Minimum
  - 1st Quartile(25th percentile)
  - Median (50th percentile)
  - Mean
  - 3rd Quartile (75th percentile)
  - Maximum

- Factors: return the frequency table

Before we summarize it, we need to make sure that all the categorical values are stored as factors in R and convert them to factors if necessary. We have converted gender and race respectively. We need to convert the other three variables to factors using the following codes:

```
StudentsPerformance$ParentalLevelOfEducation <- as.factor(StudentsPerformance
$ParentalLevelOfEducation)
StudentsPerformance$Lunch <- as.factor(StudentsPerformance$Lunch)
StudentsPerformance$TestPreparationCourse <- as.factor(StudentsPerformance$Te
stPreparationCourse)
```

Next, let's summarize the data frame using the summary() function.

```
summary(StudentsPerformance)
```