Today, I am going to cover other operators in R formula.

There is another exponent operator in R formula. The R formula

$$(V1 + V2 + V3 + \cdots + Vn)^k$$

means including **all these variables**$(V1, V2, \dots, Vn)$ and **all the interactions up to** k **way**.

For example, the following R formula:

$$MathScore \sim (ReadingScore + WritingScore + TestPreparationCourse)^3$$

includes the following items:

- Main effect: ReadingScore, WritingScore, and TestPreparationCourse
- Two way interactions: ReadingScore: WritingScore, ReadingScore:TestPreparationCourse and WritingScore:TestPreparationCourse
- Three way interactions: ReadingScore:WritingScore:TestPreparationCourse

Therefore, the following R formulas are equivalent to each other

$$MathScore \sim (ReadingScore + WritingScore + TestPreparationCourse)^3$$
$$MathScore \sim ReadingScore * WritingScore * TestPreparationCourse$$
$$\begin{aligned} MathScore \sim\ & ReadingScore + WritingScore + TestPreparationCourse + ReadingScore{:}WritingSc \\ & + ReadingScore{:}TestPreparationCourse + WritingScore{:}TestPreparationCourse \\ & + ReadingScore{:}WritingScore{:}TestPreparationCourse \end{aligned}$$

Insulation Function in R Formula

Now we are asked to build a linear model to forecast MathScore using the aggregated ReadingScore and WritingScore as a single predictor instead of two predictors. If we try to use the following R formula

$$MathScore \sim (ReadingScore + WritingScore)$$

Although there is a parenthesis there, the "+" operator in R formula means including. It still means that the Math scores depends on Readingscore and Writingscore. It is equivalent to the following math formula

$$MathScore = \beta_0 + \beta_1 * ReadingScore + \beta_2 * WritingScore$$

To let R engine treat them as a new variable, we have use the I() function/ **Insulation** function. This function will interpret any operators inside this function as **regular mathematical operators instead R formula operator**.

Let's run the following R codes;

```
lm.result6 <- lm(MathScore ~ I(ReadingScore + WritingScore), data= StudentsPe
rformance)
summary(lm.result6)
```

Therefore, we obtain the following formula

$$MathScore = 8.213934 + 0.4217592 * (ReadingScore + WritingScore)$$

1 Denotes Intercept in R Formula

The default linear models in R have the y-intercept. We may overwrite it/remove the y-intercept from the formula by using the following syntax:

$$-1$$

Note here that

- **-** means **excluding/removing** the variable in R formula
- **1** means y-intercept in the R formula

Then -1 means excluding the y-intercept.

Suppose that we believe that if the students' reading and writing scores are zeros, the math scores should also be zero. Then we can build a model using the following R formula:

$$MathScore = \beta_1 * ReadingScore + \beta_2 * WritingScore$$

Note here there is no y-intercept (or the y-intercept is 0).

We can use the following R formula to explicitly remove the y-intercept from the formula

```
lm.result7 <- lm(MathScore ~ -1+ReadingScore + WritingScore, data= StudentsPe
rformance)
summary(lm.result7)$coefficient
```

Therefore, we obtain the following math formula

$$MathScore = 0.7157145 * ReadingScore + 0.2389781 * WritingScore$$

Dot Operator in R Formula

If we want to build a R formula to include all the predictors, we can use a **.** (dot operator) to do it. It will let us write a very compact formula instead of listing all the predictors. But care must be taken, since the predictors in the formula are not explicitly showed.

We want to build a linear model to forecast the math scores using all other predictors, Gender, Race, ParentalLevelOfEducation, Lunch, TestPreparationCourse, ReadingScore, and WritingScore.

Notice that some predictors such as Gender, Race, ParentalLevelOfEducation, Lunch and TestPreparationCourse are factors/categorical variables with finite many cases. Since the R

engine can only understand factors in an R formula, we need to convert them in characters to factors using the following R codes:

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

After converting them to factors, we can build the linear model using the compact R formula with the dot operator on the right hand side of ~. It means to include all predictors, of course it **excludes the target** on the right hand size of ~.

```
lm.result8 <- lm(MathScore ~ ., data= StudentsPerformance)
summary(lm.result8)
```

Note here, there are five factors. It is not convenient for us to write down the math formula using a piece wise function with many parts. The math formula is very complicated. We skip it here.

Forecast the New Data Values

There are two main reasons on why we build models. We may want to understand the relationship between predictors and the target or forecast the target using new inputs. We have covered the first part by using the R formula and obtaining the corresponding math formula.

Let's look at the prediction of new data set. The new data set may come from external files such as in Excel files or databases. For illustration purposes, we will generate a new data frame manually:

```
new.student <- data.frame(Gender ="female", Race = "group C", ParentalLevelOf
Education = "some college",
                      Lunch = "standard", TestPreparationCourse ="none",  Read
ingScore = 70, WritingScore = 69)

predict(lm.result8, newdata = new.student)

##        1
## 62.54281
```

The new data frame must contain all the predictors with the **same** column names.

Next, we call the predict function with two arguments:

- The first argument is the model results on the training data set.
- The second argument is the new data set by specifying the new data source in data frame format

The predicted math score of this student is 62.54. It only produces a single number, which is the "average" of potential math scores.

Sometimes, we want to generate an interval estimate of the math scores. We can do it by specifying the interval argument as "predict":

```
new.student <- data.frame(Gender ="female", Race = "group C", ParentalLevelOf
Education = "some college",
                      Lunch = "standard", TestPreparationCourse ="none",  Read
ingScore = 70, WritingScore = 69)

predict(lm.result8, newdata = new.student, interval = "predict")

##        fit     lwr      upr
## 1 62.54281 51.9758 73.10983
```

It produces three numbers:

- fit value is the "average" of potential math scores
- lwr denotes the lower range of the potential math scores
- upr denotes the upper range of the potential math scores

The predicted average of math scores is 62.54.

The default of 95% confidence interval of math scores is

$$[51.98,73.11]$$

Which means if we can repeat this process for 100 times, there are at least 95 times the predicted interval containing the "actual" average.