Today, we are going to cover if conditions. We are asked to determine whether the students pass math test or not.

There are 1000 students in the data set, therefore we need to take advantage of loop. Let’s do it using for loop and if condition. Here are the R codes:

#add one column  
StudentsPerformance$MathGrade <- NULL  
for (idx in c(1:nrow(StudentsPerformance))){  
 if(StudentsPerformance[idx, "MathScore"] >= 70){  
 StudentsPerformance[idx,"MathGrade"] <- "Pass"  
 } else {  
 StudentsPerformance[idx,"MathGrade"] <- "Fail"  
 }  
}

The R codes perform the following tasks:

* Add one more column, MathGrade, to the data frame before the loop
* Loop over the rows based on the row number, generated by colon operator, c(1:nrow(StudentsPerformance)
* Inside the loop, take out the MathScore and assign the “Pass” if the math score>=70; otherwise, assign it to “Fail”.

Let’s run the codes and look at the first few rows of the data

head(StudentsPerformance)

We just used for loop and “if else”" conditions to determine the grade. There is a better solution. We only use “ifelse”" conditions without **explicitly** writing the loop.

Let’s rewrite the codes using if else

StudentsPerformance$MathGrade <- ifelse(StudentsPerformance$MathScore >=70, "Pass", "Fail")

It is easy to see that we write less code using ifelse. It runs much faster. **I strongly recommend that we use “ifelse” if possible instead of if else and loops explicitly**.

Your manager asks you to determine the students’ grades according to the following grading policy:

|  |  |
| --- | --- |
| Math Score | Math Grade |
| [90,100] | A |
| [80, 90) | B |
| [70, 80) | C |
| [0, 70) | D |

We may use nested “if else” and loops or nested “ifelse”. Let’s look at these two solutions.

Solution 1 uses nested “if else” and loops

StudentsPerformance$MathGrade <- NULL  
  
for (idx in c(1:nrow(StudentsPerformance))){  
 if(StudentsPerformance[idx, "MathScore"] >= 90){  
 StudentsPerformance[idx,"MathGrade"] <- "A"  
 } else if (StudentsPerformance[idx, "MathScore"] >= 80){  
 StudentsPerformance[idx,"MathGrade"] <- "B"  
 } else if (StudentsPerformance[idx, "MathScore"] >= 70){  
 StudentsPerformance[idx,"MathGrade"] <- "C"  
 } else {  
 StudentsPerformance[idx,"MathGrade"] <- "D"  
 }  
}

The R codes perform the following tasks:

* Add one more column, MathGrade, to the data frame before the loop and initialize to null values
* Loop over the rows based on the row number, generated by colon operator, c(1:nrow(StudentsPerformance)
* Inside the loop, take out the MathScore and assign the corresponding grades.
  + If MathScore >=90; the grade is A
  + Else automatically covers the negation of Mathsocre <90 and combines MathScore>=80; which means 80 <= MathScore<90; note here we only write one condition Mathscore>=80; the R engine automatically knows that MathScore< 90 because of the “else”. Please **DON’T** write the (MathScore < 90) &( MathScore <=80) since it is not efficient and runs slower.
  + …..
  + The last “else” covers the negation of MathScore >= 70, which means MathScore < 70 since R engine knows it, we don’t need to write this condition explicitly.

A better solution is to use nested ifelse.

StudentsPerformance$MathGrade <- ifelse(StudentsPerformance$MathScore >=90, "A",ifelse(StudentsPerformance$MathScore >=80, "B", ifelse(StudentsPerformance$MathScore >=70, "C", "D")))

The nested “ifelse” works similarly but without using the loop explicitly.

Let’s look at last few records:

tail(StudentsPerformance)

Again, when we have a choice to use either if and loops or “ifelse”, we should use “ifelse”. There are two advantages:

* The R codes are more compact, therefore it is easier for us to understand and maintain.
* It typically runs much faster due to vectorization.