I am going to attempt to answer this question — but this is not easy, and it is a question I have avoided for a long time.  This is me putting on a training cap to see if I can properly answer it.

When you are dealing with situations like this:

  if(lowTemps[i] > 40 & highTemps[i] < 60 & noonCond[i] == "Sunny")

  {

    cat("Day", i, "meets these conditions \n");

  }

**&** will evaluate all three conditions (lowTemp, highTemp, and noonCond) — even if there is no need.  For instance, if ***lowTemp[i] is 30*** then the first condition fails.  If the first condition fails using ***&***, the whole condition automatically fails.  There is technically no need to check ***highTemps***and ***noonCond***.  But, **&**will soldier on and dutifully check the other conditions.

**&&**will be lazy and not check the other conditions since the answer is already known.  So, **&&** is a little more efficient.

**Why would you want to use & if && is more efficient?**

*This is a rabbit hole that you probably will not encounter in your coding (I have encountered this a few times in my coding — but that's a few times in 20 years).*

Because sometimes programmers are doing weird things inside the if statement, like this:

  if(lowTemps[i] > 40 & highTemps[i] < 60 & (sunnyDay = noonCond[i] == "Sunny") )

  {

    cat("Day", i, "meets these conditions \n");

  }

In this case, the ***if***statement is doing more than just checking conditions, it is setting the variable ***sunnyDay***, which will equal **TRUE**or **FALSE**.

***sunnyDay***will be set if you use **&** but might not be set if you use **&&**.

if ***lowTemps[i] is 30***, **&&**will quit right there, where **&** will soldier on and complete the whole ***if*** statement.

Using ***&***: If the variable ***temp***does not exist, then ***temp > 50*** will throw an error

(exists("temp") & temp > 50)  # will give an error if temp does not exist

Using &&: If the variable ***temp***does not exist, this will evaluate to FALSE and not check if ***temp>50***

(exists("temp") && temp > 50) # will not give an error

This is something I use a lot when I program webpages and do not know what inputs the user has provided.

**& can be use to compare vectors, && cannot:**

The other difference is if you have Boolean vectors you are comparing like these:

sunnyDays = c(TRUE, TRUE, FALSE, FALSE)

hotDays = c(TRUE, FALSE, TRUE, FALSE)

***&*** allows you to compare vectors, whereas ***&&*** will give you an error

sunnyDays & hotDays;    # c(TRUE, FALSE, FALSE, FALSE)

sunnyDays && hotDay;    # error

& compares each value in the vectors so the in-between steps looks like this:

sunnyDays & hotDays;

c(sunnyDays[1]& hotDays[1],

  sunnyDays[2]& hotDays[2],

  sunnyDays[3]& hotDays[3],

  sunnyDays[4]& hotDays[4])

c(TRUE&TRUE,

  TRUE&FALSE,

  FALSE&TRUE,

  FALSE&FALSE)

c(TRUE, FALSE, FALSE, FALSE)