|  |
| --- |
| title: "DataCamp - Intermediate R" |
| author: "[Luka Ignjatovi&#263;](https://github.com/LukaIgnjatovic)" |
| output: |
| html\_document: |
| highlight: tango |
| theme: united |
| toc: yes |
| toc\_depth: 4 |
| keep\_md: true |
| md\_document: |
| toc: true |
| toc\_depth: 4 |
| --- |
|  |
| **## Conditionals and control flow** |
|  |
| To be TRUE or not be TRUE, that's the question. In this chapter you'll learn about relational operators to see how R objects compare and logical operators to combine logicals. Next, you'll use this knowledge to build conditional statements. |
|  |
| <div align="middle"> |
|  |
| > \*\*Document:\*\* ["Slides - Conditional and control flows"](./Slides/Chapter 01 - Conditional and control flows.pdf) |
|  |
| </div> |
|  |
| <div align="middle"> |
|  |
| <video width="80%" controls src="./Videos/Chapter 01 - Lecture 01 - Relational operators.mp4" type="video/mp4"/> |
|  |
| </div> |
|  |
| **### Equality** |
|  |
| The most basic form of comparison is equality. Let's briefly recap its syntax. The following statements all evaluate to `TRUE` (feel free to try them out in the console). |
|  |
| 3 == (2 + 1) |
| "intermediate" != "r" |
| TRUE != FALSE |
| "Rchitect" != "rchitect" |
|  |
| Notice from the last expression that R is case sensitive: "R" is not equal to "r". Keep this in mind when solving the exercises in this chapter! |
|  |
| **#### Instructions** |
| \* *\*In the editor on the right, write R code to see if\** `TRUE` *\*equals\** `FALSE`*\*.\** |
| \* *\*Likewise, check if\** `-6 \* 14` *\*is not equal to\** `17 - 101`*\*.\** |
| \* *\*Next up: comparison of character strings. Ask R whether the strings\** `"useR"` *\*and\** `"user"` *\*are equal.\** |
| \* *\*Finally, find out what happens if you compare logicals to numerics: are\** `TRUE` *\*and\** `1` *\*equal?\** |
| ```{r} |
| # Comparison of logicals |
| TRUE == FALSE |
|  |
| # Comparison of numerics |
| -6 \* 14 != 17 - 101 |
|  |
| # Comparison of character strings |
| "useR" == "user" |
|  |
| # Compare a logical with a numeric |
| TRUE == 1 |
| ``` |
|  |
| **\*\*Awesome! Since\*\*** `TRUE` **\*\*coerces to\*\*** `1` **\*\*under the hood,\*\*** `TRUE == 1` **\*\*evaluates to\*\*** `TRUE`**\*\*. Make sure not to mix up\*\*** `==` **\*\*(comparison) and\*\*** `=` **\*\*(assignment),\*\*** `==` **\*\*is what need to check the equality of R objects.\*\*** |
|  |
| **### Greater and less than** |
|  |
| Apart from equality operators, Filip also introduced the less than and greater than operators: `<` and `>`. You can also add an equal sign to express less than or equal to or greater than or equal to, respectively. Have a look at the following R expressions, that all evaluate to `FALSE`: |
|  |
| (1 + 2) > 4 |
| "dog" < "Cats" |
| TRUE <= FALSE |
|  |
| Remember that for string comparison, R determines the greater than relationship based on alphabetical order. Also, keep in mind that `TRUE` corresponds to `1` in R, and `FALSE` coerces to `0` behind the scenes. Therefore, `FALSE < TRUE` is `TRUE`. |
|  |
| **#### Instructions** |
| *\*Write R expressions to check whether:\** |
|  |
| \* `-6 \* 5 + 2` *\*is greater than or equal to\** `-10 + 1`*\*.\** |
| \* `"raining"`\* is less than or equal to\* `"raining dogs"`*\*.\** |
| \* `TRUE` *\*is greater than\** `FALSE`*\*.\** |
| ```{r} |
| # Comparison of numerics |
| -6 \* 5 + 2 >= -10 + 1 |
|  |
| # Comparison of character strings |
| "raining" <= "raining dogs" |
|  |
| # Comparison of logicals |
| TRUE > FALSE |
| ``` |
|  |
| **\*\*Great job! Make sure to have a look at the console output to see if R returns the results you expected.\*\*** |
|  |
| **### Compare vectors** |
|  |
| You are already aware that R is very good with vectors. Without having to change anything about the syntax, R's relational operators also work on vectors. |
|  |
| Let's go back to the example that was started in the video. You want to figure out whether your activity on social media platforms have paid off and decide to look at your results for LinkedIn and Facebook. The sample code in the editor initializes the vectors `linkedin` and `facebook`. Each of the vectors contains the number of profile views your LinkedIn and Facebook profiles had over the last seven days. |
|  |
| **#### Instructions** |
| *\*Using relational operators, find a logical answer, i.e.\** `TRUE` *\*or\** `FALSE`*\*, for the following questions:\** |
|  |
| \* *\*On which days did the number of LinkedIn profile views exceed 15?\** |
| \* *\*When was your LinkedIn profile viewed only 5 times or fewer?\** |
| \* *\*When was your LinkedIn profile visited more often than your Facebook profile?\** |
| ```{r} |
| # The linkedin and facebook vectors have already been created for you |
| linkedin <- c(16, 9, 13, 5, 2, 17, 14) |
| facebook <- c(17, 7, 5, 16, 8, 13, 14) |
|  |
| # Popular days |
| linkedin > 15 |
|  |
| # Quiet days |
| linkedin <= 5 |
|  |
| # LinkedIn more popular than Facebook |
| linkedin > facebook |
| ``` |
|  |
| **\*\*Wonderful! Have a look at the console output. Your LinkedIn profile was pretty popular on the sixth day, but less so on the fourth and fifth day.\*\*** |
|  |
| **### Compare matrices** |
|  |
| R's ability to deal with different data structures for comparisons does not stop at vectors. Matrices and relational operators also work together seamlessly! |
|  |
| Instead of in vectors (as in the previous exercise), the LinkedIn and Facebook data is now stored in a matrix called `views`. The first row contains the LinkedIn information; the second row the Facebook information. The original vectors `facebook` and `linkedin` are still available as well. |
|  |
| **#### Instructions** |
| *\*Using the relational operators you've learned so far, try to discover the following:\** |
|  |
| \* *\*When were the views exactly equal to 13? Use the\** `views` *\*matrix to return a logical matrix.\** |
| \* *\*For which days were the number of views less than or equal to 14? Again, have R return a logical matrix.\** |
| ```{r} |
| # The social data has been created for you |
| linkedin <- c(16, 9, 13, 5, 2, 17, 14) |
| facebook <- c(17, 7, 5, 16, 8, 13, 14) |
| views <- matrix(c(linkedin, facebook), nrow = 2, byrow = TRUE) |
|  |
| # When does views equal 13? |
| views == 13 |
|  |
| # When is views less than or equal to 14? |
| views <= 14 |
| ``` |
|  |
| **\*\*Nice job! This exercise concludes the part on comparators. Now that you know how to query the relation between R objects, the next step will be to use the results to alter the behavior of your programs. Find out all about that in the next video!\*\*** |
|  |
| <div align="middle"> |
|  |
| <video width="80%" controls src="./Videos/Chapter 01 - Lecture 02 - Logical operators.mp4" type="video/mp4" /> |
|  |
| </div> |
|  |
| **### & and |** |
|  |
| Before you work your way through the next exercises, have a look at the following R expressions. All of them will evaluate to `TRUE`: |
|  |
| TRUE & TRUE |
| FALSE | TRUE |
| 5 <= 5 & 2 < 3 |
| 3 < 4 | 7 < 6 |
|  |
| Watch out: `3 < x < 7` to check if `x` is between 3 and 7 will not work; you'll need `3 < x & x < 7` for that. |
|  |
| In this exercise, you'll be working with the `last` variable. This variable equals the last value of the `linkedin` vector that you've worked with previously. The `linkedin` vector represents the number of LinkedIn views your profile had in the last seven days, remember? Both the variables `linkedin` and `last` have already been defined in the editor. |
|  |
| **#### Instructions** |
| *\*Write R expressions to solve the following questions concerning the variable\** `last`*\*:\** |
|  |
| \* *\*Is\** `last` *\*under 5 or above 10?\** |
| \* *\*Is\** `last` *\*between 15 and 20, excluding 15 but including 20?\** |
| ```{r} |
| # The linkedin and last variable are already defined for you |
| linkedin <- c(16, 9, 13, 5, 2, 17, 14) |
| last <- tail(linkedin, 1) |
|  |
| # Is last under 5 or above 10? |
| last < 5 | last > 10 |
|  |
| # Is last between 15 (exclusive) and 20 (inclusive)? |
| last > 15 & last <= 20 |
| ``` |
|  |
| **\*\*Great! Have one last look at the console before proceeding; do the results of the different expressions make sense?\*\*** |
|  |
| **### & and | (2)** |
|  |
| Like relational operators, logical operators work perfectly fine with vectors and matrices. |
|  |
| Both the vectors `linkedin` and `facebook` are available again. Also a matrix - `views` - has been defined; its first and second row correspond to the `linkedin` and `facebook` vectors, respectively. Ready for some advanced queries to gain more insights into your social outreach? |
|  |
| **#### Instructions** |
| \* *\*When did LinkedIn views exceed 10 and did Facebook views fail to reach 10 for a particular day? Use the\** `linkedin` *\*and\** `facebook` *\*vectors.\** |
| \* *\*When were one or both of your LinkedIn and Facebook profiles visited at least 12 times?\** |
| \* *\*When is the\** `views` *\*matrix equal to a number between 11 and 14, excluding 11 and including 14?\** |
| ```{r} |
| # The social data (linkedin, facebook, views) has been created for you |
|  |
| # linkedin exceeds 10 but facebook below 10 |
| linkedin > 10 & facebook < 10 |
|  |
| # When were one or both visited at least 12 times? |
| linkedin >= 12 | facebook >= 12 |
|  |
| # When is views between 11 (exclusive) and 14 (inclusive)? |
| views > 11 & views <= 14 |
| ``` |
|  |
| **\*\*Bravo! You'll have noticed how easy it is to use logical operators to vectors and matrices. What do these results tell us? The third day of the recordings was the only day where your LinkedIn profile was visited more than 10 times, while your Facebook profile wasn't. Can you draw similar conclusions for the other results?\*\*** |
|  |
| **### Reverse the result: !** |
|  |
| On top of the `&` and `|` operators, you also learned about the `!` operator, which negates a logical value. To refresh your memory, here are some R expressions that use `!`. They all evaluate to `FALSE`: |
|  |
| !TRUE |
| !(5 > 3) |
| !!FALSE |
|  |
| What would the following set of R expressions return? |
|  |
| x <- 5 |
| y <- 7 |
| !(!(x < 4) & !!!(y > 12)) |
|  |
| *\*Possible answers:\** |
|  |
| \* *\*TRUE\** |
| \* **\*\*FALSE\*\*** |
| \* *\*Running this piece of code would throw an error.\** |
|  |
| **\*\*Great!\*\*** |
|  |
| **### Blend it all together** |
|  |
| With the things you've learned by now, you're able to solve pretty cool problems. |
|  |
| Instead of recording the number of views for your own LinkedIn profile, suppose you conducted a survey inside the company you're working for. You've asked every employee with a LinkedIn profile how many visits their profile has had over the past seven days. You stored the results in a data frame called `li\_df`. This data frame is available in the workspace; type `li\_df` in the console to check it out. |
|  |
| **#### Instructions** |
| \* *\*Select the entire second column, named\** `day2`*\*, from the\** `li\_df` *\*data frame as a vector and assign it to\** `second`*\*.\** |
| \* *\*Use\** `second` *\*to create a logical vector, that contains\** `TRUE` *\*if the corresponding number of views is strictly greater than 25 or strictly lower than 5 and\** `FALSE` *\*otherwise. Store this logical vector as\** `extremes`*\*.\** |
| \* *\*Use\** `sum()` *\*on the\** `extremes` *\*vector to calculate the number of\** `TRUE`*\*s in extremes (i.e. to calculate the number of employees that are either very popular or very low-profile). Simply print this number to the console.\** |
| ```{r} |
| # Constructing the li\_df data frame |
| employee\_1 <- c(2, 3, 3, 6, 4, 2, 0) |
| employee\_2 <- c(19, 23, 18, 22, 23, 29, 25) |
| employee\_3 <- c(24, 18, 15, 19, 18, 22, 17) |
| employee\_4 <- c(22, 18, 27, 26, 19, 21, 25) |
| employee\_5 <- c(25, 25, 26, 31, 24, 36, 37) |
| employee\_6 <- c(22, 20, 29, 26, 23, 22, 29) |
| employee\_7 <- c(0, 4, 2, 2, 3, 4, 2) |
| employee\_8 <- c(12, 3, 15, 7, 1, 15, 11) |
| employee\_9 <- c(19, 22, 22, 19, 25, 24, 23) |
| employee\_10 <- c(23, 12, 19, 25, 18, 22, 22) |
| employee\_11 <- c(29, 27, 23, 25, 29, 30, 17) |
| employee\_12 <- c(13, 13, 20, 17, 12, 22, 20) |
| employee\_13 <- c(7, 17, 9, 5, 11, 9, 9) |
| employee\_14 <- c(26, 27, 28, 36, 29, 31, 30) |
| employee\_15 <- c(7, 6, 4, 11, 5, 5, 15) |
| employee\_16 <- c(32, 35, 31, 35, 24, 25, 36) |
| employee\_17 <- c(7, 17, 9, 12, 13, 6, 12) |
| employee\_18 <- c(9, 6, 3, 12, 3, 8, 6) |
| employee\_19 <- c(0, 1, 11, 6, 0, 4, 11) |
| employee\_20 <- c(9, 12, 6, 13, 12, 13, 11) |
| employee\_21 <- c(6, 15, 15, 10, 9, 7, 18) |
| employee\_22 <- c(17, 17, 12, 4, 14, 17, 7) |
| employee\_23 <- c(1, 12, 8, 2, 4, 4, 11) |
| employee\_24 <- c(5, 8, 0, 1, 6, 3, 1) |
| employee\_25 <- c(2, 7, 5, 3, 1, 5, 5) |
| employee\_26 <- c(29, 25, 32, 28, 28, 27, 27) |
| employee\_27 <- c(17, 15, 17, 23, 23, 17, 22) |
| employee\_28 <- c(26, 32, 33, 30, 33, 28, 26) |
| employee\_29 <- c(27, 29, 24, 29, 26, 31, 28) |
| employee\_30 <- c(4, 1, 1, 2, 1, 7, 4) |
| employee\_31 <- c(22, 22, 17, 20, 14, 19, 13) |
| employee\_32 <- c(9, 11, 7, 10, 8, 15, 5) |
| employee\_33 <- c(6, 5, 12, 5, 17, 17, 4) |
| employee\_34 <- c(18, 17, 12, 22, 22, 13, 12) |
| employee\_35 <- c(2, 12, 13, 7, 10, 6, 2) |
| employee\_36 <- c(32, 26, 20, 23, 24, 25, 21) |
| employee\_37 <- c(5, 13, 12, 11, 6, 5, 10) |
| employee\_38 <- c(6, 10, 11, 6, 6, 2, 5) |
| employee\_39 <- c(30, 37, 32, 35, 37, 41, 42) |
| employee\_40 <- c(34, 33, 32, 35, 33, 27, 35) |
| employee\_41 <- c(15, 19, 21, 18, 22, 26, 22) |
| employee\_42 <- c(28, 29, 30, 19, 21, 19, 26) |
| employee\_43 <- c(6, 8, 6, 7, 17, 11, 14) |
| employee\_44 <- c(17, 22, 27, 24, 18, 28, 24) |
| employee\_45 <- c(6, 10, 17, 18, 13, 10, 7) |
| employee\_46 <- c(18, 19, 22, 17, 21, 15, 23) |
| employee\_47 <- c(21, 27, 28, 28, 26, 17, 25) |
| employee\_48 <- c(10, 18, 20, 18, 12, 19, 17) |
| employee\_49 <- c(6, 15, 15, 15, 10, 14, 2) |
| employee\_50 <- c(30, 28, 29, 31, 24, 20, 25) |
|  |
| li\_df <- c(employee\_1, employee\_2, employee\_3, employee\_4, employee\_5, |
| employee\_6, employee\_7, employee\_8, employee\_9, employee\_10, |
| employee\_11, employee\_12, employee\_13, employee\_14, employee\_15, |
| employee\_16, employee\_17, employee\_18, employee\_19, employee\_20, |
| employee\_21, employee\_22, employee\_23, employee\_24, employee\_25, |
| employee\_26, employee\_27, employee\_28, employee\_29, employee\_30, |
| employee\_31, employee\_32, employee\_33, employee\_34, employee\_35, |
| employee\_36, employee\_37, employee\_38, employee\_39, employee\_40, |
| employee\_41, employee\_42, employee\_43, employee\_44, employee\_45, |
| employee\_46, employee\_47, employee\_48, employee\_49, employee\_50) |
|  |
| li\_df <- matrix(li\_df, nrow = 50, byrow = TRUE) |
|  |
| colnames(li\_df) <- c("day1", "day2", "day3", "day4", "day5", "day6", "day7") |
|  |
| li\_df <- data.frame(li\_df) |
|  |
| # Select the second column, named day2, from li\_df: second |
| second <- li\_df[, 2] |
|  |
| # Build a logical vector, TRUE if value in second is extreme: extremes |
| extremes <- second > 25 | second < 5 |
|  |
| # Count the number of TRUEs in extremes |
| sum(extremes) |
|  |
| # Solve it with a one-liner |
| sum(li\_df[, 2] > 25 | li\_df[, 2] < 5) |
| ``` |
|  |
| **\*\*Great! Head over to the next video and learn how relational and logical operators can be used to alter the flow of your R scripts.\*\*** |
|  |
| <div align="middle"> |
|  |
| <video width="80%" controls src="./Videos/Chapter 01 - Lecture 03 - Conditional statements.mp4" type="video/mp4" /> |
|  |
| </div> |
|  |
| **### The if statement** |
|  |
| Before diving into some exercises on the `if` statement, have another look at its syntax: |
|  |
| if (condition) { |
| expr |
| } |
|  |
| Remember your vectors with social profile views? Let's look at it from another angle. The `medium` variable gives information about the social website; the `num\_views` variable denotes the actual number of views that particular `medium` had on the last day of your recordings. Both these variables have already been defined in the editor. |
|  |
| **#### Instructions** |
| \* *\*Examine the\** `if` *\*statement that prints out "Showing LinkedIn information" if the\** `medium` *\*variable equals "LinkedIn".\** |
| \* *\*Code an\** `if` *\*statement that prints "You're popular!" to the console if the num\_views variable exceeds 15.\** |
| ```{r} |
| # Variables related to your last day of recordings |
| medium <- "LinkedIn" |
| num\_views <- 14 |
|  |
| # Examine the if statement for medium |
| if (medium == "LinkedIn") { |
| print("Showing LinkedIn information") |
| } |
|  |
| # Write the if statement for num\_views |
| if (num\_views > 15) { |
| print("You're popular!") |
| } |
| ``` |
|  |
| **\*\*Great! Try to see what happens if you change the\*\*** `medium` **\*\*and\*\*** `num\_views` **\*\*variables and run your code again. Let's further customize these if statements in the next exercise.\*\*** |
|  |
| **### Add an else** |
|  |
| You can only use an `else` statement in combination with an `if` statement. The `else` statement does not require a condition; its corresponding code is simply run if all of the preceding conditions in the control structure are `FALSE`. Here's a recipe for its usage: |
|  |
| if (condition) { |
| expr1 |
| } else { |
| expr2 |
| } |
|  |
| *\*It's important that the\** `else` *\*keyword comes on the same line as the closing bracket of the\** `if` *\*part!\** |
|  |
| Both `if` statements that you coded in the previous exercises are already available in the editor. It's now up to you to extend them with the appropriate `else` statements! |
|  |
| **#### Instructions** |
| *\*Add an else statement to both control structures, such that:\** |
|  |
| \* *\*"Unknown medium" gets printed out to the console when the if-condition on\** `medium` *\*does not hold.\** |
| \* *\*R prints out "Try to be more visible!" when the if-condition on\** `num\_views` *\*is not met.\** |
| ```{r} |
| # Variables related to your last day of recordings |
| medium <- "LinkedIn" |
| num\_views <- 14 |
|  |
| # Control structure for medium |
| if (medium == "LinkedIn") { |
| print("Showing LinkedIn information") |
| } else { |
| print("Unknown medium") |
| } |
|  |
| # Control structure for num\_views |
| if (num\_views > 15) { |
| print("You're popular!") |
| } else { |
| print("Try to be more visible!") |
| } |
| ``` |
|  |
| **\*\*Great job! You also had Facebook information available, remember? Time to add some more statements to our control structures using\*\*** `else if`**\*\*!\*\*** |
|  |
| **### Customize further: else if** |
|  |
| The `else if` statement allows you to further customize your control structure. You can add as many `else if` statements as you like. Keep in mind that R ignores the remainder of the control structure once a condition has been found that is `TRUE` and the corresponding expressions have been executed. Here's an overview of the syntax to freshen your memory: |
|  |
| if (condition1) { |
| expr1 |
| } else if (condition2) { |
| expr2 |
| } else if (condition3) { |
| expr3 |
| } else { |
| expr4 |
| } |
|  |
| *\*Again, it's important that the\** `else if` *\*keywords comes on the same line as the closing bracket of the previous part of the control construct!\** |
|  |
| **#### Instructions** |
| *\*Add code to both control structures such that:\** |
|  |
| \* *\*R prints out "Showing Facebook information" if\** `medium` *\*is equal to "Facebook". Remember that R is case sensitive!\** |
| \* *\*"Your number of views is average" is printed if\** `num\_views` *\*is between 15 (inclusive) and 10 (exclusive). Feel free to change the variables\** `medium` *\*and\** `num\_views` *\*to see how the control structure respond. In both cases, the existing code should be extended in the\** `else if` *\*statement. No existing code should be modified.\** |
| ```{r} |
| # Variables related to your last day of recordings |
| medium <- "LinkedIn" |
| num\_views <- 14 |
|  |
| # Control structure for medium |
| if (medium == "LinkedIn") { |
| print("Showing LinkedIn information") |
| } else if (medium == "Facebook") { |
| # Add code to print correct string when condition is TRUE |
| print("Showing Facebook information") |
| } else { |
| print("Unknown medium") |
| } |
|  |
| # Control structure for num\_views |
| if (num\_views > 15) { |
| print("You're popular!") |
| } else if (num\_views <= 15 & num\_views > 10) { |
| # Add code to print correct string when condition is TRUE |
| print("Your number of views is average") |
| } else { |
| print("Try to be more visible!") |
| } |
| ``` |
|  |
| **\*\*Awesome! Have another look at the second control structure. Because R abandons the control flow as soon as it finds a condition that is met, you can simplify the condition for the\*\*** `else if` **\*\*part in the second construct to\*\*** `num\_views > 10`**\*\*.\*\*** |
|  |
| **### Else if 2.0** |
|  |
| You can do anything you want inside if-else constructs. You can even put in another set of conditional statements. Examine the following code chunk: |
|  |
| if (number < 10) { |
| if (number < 5) { |
| result <- "extra small" |
| } else { |
| result <- "small" |
| } |
| } else if (number < 100) { |
| result <- "medium" |
| } else { |
| result <- "large" |
| } |
| print(result) |
|  |
| Have a look at the following statements: |
|  |
| 1. If `number` is set to 6, "small" gets printed to the console. |
| 2. If `number` is set to 100, R prints out "medium". |
| 3. If `number` is set to 4, "extra small" gets printed out to the console. |
| 4. If `number` is set to 2500, R will generate an error, as `result` will not be defined. |
|  |
| Select the option that lists all the true statements. |
|  |
| *\*Possible answers:\** |
|  |
| \* *\*(2) and (4)\** |
| \* *\*(1) and (4)\** |
| \* **\*\*(1) and (3)\*\*** |
| \* *\*(2) and (3)\** |
|  |
| **\*\*Wonderful! If you got this one right, the next exercise will be a piece of cake.\*\*** |
|  |
| **### Take control!** |
|  |
| In this exercise, you will combine everything that you've learned so far: relational operators, logical operators and control constructs. You'll need it all! |
|  |
| In the editor, we've coded two values beforehand: `li` and `fb`, denoting the number of profile views your LinkedIn and Facebook profile had on the last day of recordings. Go through the instructions to create R code that generates a 'social media score', `sms`, based on the values of `li` and `fb`. |
|  |
| **#### Instructions** |
| *\*Finish the control-flow construct with the following behavior:\** |
|  |
| \* *\*If both\** `li` *\*and\** `fb` *\*are 15 or higher, set\** `sms` *\*equal to double the sum of\** `li` *\*and\** `fb`*\*.\** |
| \* *\*If both\** `li` *\*and\** `fb` *\*are strictly below 10, set\** `sms` *\*equal to half the sum of\** `li` *\*and\** `fb`*\*.\** |
| \* *\*In all other cases, set\** `sms` *\*equal to\** `li + fb`*\*.\** |
| \* *\*Finally, print the resulting\** `sms` *\*variable to the console.\** |
| ```{r} |
| # Variables related to your last day of recordings |
| li <- 15 |
| fb <- 9 |
|  |
| # Code the control-flow construct |
| if (li >= 15 & fb >= 15) { |
| sms <- 2 \* (li + fb) |
| } else if (li < 10 & fb < 10) { |
| sms <- 0.5 \* (li + fb) |
| } else { |
| sms <- li + fb |
| } |
|  |
| # Print the resulting sms to the console |
| sms |
| ``` |
|  |
| **\*\*Bellissimo! Feel free to play around some more with your solution by changing the values of\*\*** `li` **\*\*and\*\*** `fb`**\*\*.\*\*** |
| **\*\*You have finished the chapter "Conditionals and control flow"!\*\*** |