Assignment 12: Conditionals

Ellen Bledsoe

2025-05-15

Assignment

1. Choice Operators (20 pts)

Create the following variables by running the code chunk below.

```
w <- 10.2
x <- 1.3
y <- 2.8
z <- 17.5
colors <- c("red", "blue", "green")
masses <- c(45.2, 36.1, 27.8, 81.6, 42.4)
dna1 <- "attattaggaccaca"
dna2 <- "attattaggaacaca"
```

Using those variables, write code that determines whether the following statements are `TRUE` or `FALSE`.

a. `w` is greater than 10
b. `"green"` is in `colors`
c. `x` is greater than `y`
d. Each value in `masses` is greater than 40.
e. `dna1` is the same as `dna2`
f. `dna1` is not the same as `dna2`
g. `w` is greater than `x`, or `y` is greater than `z`
h. `x` times `w` is between 13.2 and 13.5
i. Each mass in `masses` is between 30 and 50.

```
### Code solution for Choice Operators

# a.  `w` is greater than 10
w > 10
```

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

```
# b.  "green" is in `colors`
colors == "green"
```

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

```
# c.  `x` is greater than `y`
x > y
```

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

```
# d. Each value in `masses` is greater than 50.
masses > 40
```

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

```
# e.  `dna1` is the same as `dna2`
dna1 == dna2
```

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

```
# f.  `dna1` is not the same as `dna2`
dna1 != dna2
```

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

```
# g. `w` is greater than `x`, or `y` is greater than `z`
w > x | y > z
```

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

```
# h.  `x` times `w` is between 13.2 and 13.5
13.2 < x * w & x * w < 13.5  # | (a pipe) will return TRUE for all values.
```

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

```
# i. Each mass in `masses` is between 30 and 50.
masses > 30 & masses < 50
```

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

2. If Statements (20 points)

a. Complete the following `if` statement so that if `age_class` is equal to "sapling" it sets `y <- 10`.

```
age_class = "sapling"
if (){

}
y
```

b. Complete the following `if` statement so that if `age_class` is equal to "sapling" it sets `y <- 10` and if `age_class` is equal to "seedling" it sets `y <- 5`.

```
age_class = "seedling"
if (){

}
y
```

c. Complete the following `if` statement so that if `age_class` is equal to "sapling" it sets `y <- 10` and if `age_class` is equal to "seedling" it sets `y <- 5` and if `age_class` is something else then it sets the value of `y <- 0`.

```
age_class = "adult"
if (){

}
y
```

d. Convert your conditional statement from (c) into a function that takes `age_class` as an argument and returns `y`. Call this function 5 times, once with each of the following values for `age_class`: "sapling", "seedling", "adult", "mature", "established".

```
# a. Complete (i.e., copy into your code and them modify) the following `if`
#   statement so that if `age_class` is equal to "sapling" it sets `y <- 10`.
```

```r
age_class = "sapling"
if (age_class == "sapling"){
  y <-10
}
y
```

```
## [1] 10
```

```r
# b. Complete the following `if` statement so that if `age_class` is equal to
#   "sapling" it sets `y <- 10` and if `age_class` is equal to "seedling" it
#   sets `y <- 5`.
age_class = "seedling"
if (age_class == "sapling"){
  y <-10
} else if (age_class == "seedling"){
  y <- 5
}
y
```

```
## [1] 5
```

```r
# c. Complete the following `if` statement so that if `age_class` is equal to
#   "sapling" it sets `y <- 10` and if `age_class` is equal to "seedling" it
#   sets `y <- 5` and if `age_class` is something else then it sets the value of
#   `y <- 0`.
age_class = "adult"
if (age_class == "sapling"){
  y <-10
} else if (age_class == "seedling") {
  y <- 5
} else {
  y <- 0
}
y
```

```
## [1] 0
```

```r
# d. Convert your conditional statement from (c) into a function that takes
#   `age_class` as an argument and returns `y`. Call this function 5 times, once
#   with each of the following values for `age_class`: "sapling", "seedling",
#   "adult", "mature", "established".
get_y_from_age_class <- function(age_class){
  if (age_class == "sapling"){
    y <-10
  } else if (age_class == "seedling") {
    y <- 5
  } else {
    y <- 0
  }
  return(y)
}

get_y_from_age_class("sapling")
```

```
## [1] 10
```

```
get_y_from_age_class("seedling")
```

```
## [1] 5
```

```
get_y_from_age_class("adult")
```

```
## [1] 0
```

```
get_y_from_age_class("mature")
```

```
## [1] 0
```

```
get_y_from_age_class("established")
```

```
## [1] 0
```

3. If Statements in Functions (20 points)

a. Write a function named `double_if_small` that takes a number as input and returns the number multiplied by 2 if the input is less than 26 and returns just the number (not multiplied by two) if the input is greater than or equal to 26.
b. Call the function from (b) with `10` as the input.
c. Call the function from (b) with `30` as the input.
d. Write a function called `prediction` that takes a single argument `x`. If `x` is both greater than 0 and less than 15 then return `y = 6 + 0.8 * x`. If `x` is both greater than 15 and less than 30 then return `y = 5 + 0.75 * x`. In all other cases return `y = NA`.
e. Call the function from (d) with `5` as the input.
f. Call the function from (d) with `26` as the input.
g. Call the function from (d) with `-2` as the input.

```
# Solution to If Statements in Functions exercise

# 3a
double_if_small <- function(number){
  if (number < 26){
    output = number * 2
  } else {
    output = number
  }
  return(output)
}

# 3b
double_if_small(10)
```

```
## [1] 20
```

```
#3c
double_if_small(30)
```

```
## [1] 30
```

```
#3d

prediction <- function(x){
  if (x > 0 & x < 15){
      y = 6 + 0.8 * x
  } else if (x > 15 & x < 30) {
```

```
      y = 5 + 0.75 * x
  } else {
      y = NA
  }
  return (y)
}

print("3e")
```

```
## [1] "3e"
```

```
prediction(5)
```

```
## [1] 10
```

```
print("3f")
```

```
## [1] "3f"
```

```
prediction(26)
```

```
## [1] 24.5
```

```
print("3g")
```

```
## [1] "3g"
```

```
prediction(-2)
```

```
## [1] NA
```

4. Size Estimates by Name (20 points)

This is a follow up to "Use and Modify" from Assignment 10.

To make it even easier to work with your dinosaur size estimation functions, you decide to create a function that lets you specify which dinosaur group you need to estimate the size of by name and then have the function automatically choose the right parameters.

Remember, the general form of the equation is:

mass <- a * length ^ b

Create a new function `get_mass_from_length_by_name()` that takes two arguments: the `length` and the name of the dinosaur group.

Inside this function use `if`/`else if`/`else` statements to check to see if the name is one of the following values and if so use the associated `a` and `b` values to estimate the species mass using these equations:

- *Stegosauria*: `mass = 10.95 * length ^ 2.64` (Seebacher 2001)
- *Theropoda*: `mass = 0.73 * length ^ 3.63` (Seebacher 2001)
- *Sauropoda*: `mass = 214.44 * length ^ 1.46` (Seebacher 2001)

If the name is not any of these values the function should return `NA`.

Run the function for:

a. A *Stegosauria* that is 10 meters long.
b. A *Theropoda* that is 8 meters long.
c. A *Sauropoda* that is 12 meters long.
d. An *Ankylosauria* that is 13 meters long.

```r
get_mass_from_length <- function(a, b, length){
  mass = a * length ** b
  return (mass)
}

get_mass_from_length_by_name <- function(length, name){
  if (name == "Stegosauria"){
    if (length > 8){
      a = 10.95
      b = 2.64
    } else {
      a = 8.5
      b = 2.8
    }
  }
  else if (name == "Theropoda"){
    a = 0.73
    b = 3.63
  }
  else if (name == "Sauropoda"){
    a = 214.44
    b = 1.46
  }
  else {
    return(NA)
  }
  get_mass_from_length(a, b, length)
}

get_mass_from_length_by_name(10, "Stegosauria")
```

```
## [1] 4779.848
```

```r
get_mass_from_length_by_name(8, "Theropoda")
```

```
## [1] 1385.286
```

```r
get_mass_from_length_by_name(12, "Sauropoda")
```

```
## [1] 8070.685
```

```r
get_mass_from_length_by_name(13, 'Ankylosauria')
```

```
## [1] NA
```

Challenge 1 (optional): If the name is not one of values that have a and b values print warning that it doesn't know how to convert that group that includes that groups name in a message like "No known estimation for Ankylosauria." You can use the function `warning("your warning text")` to print a warning and the function `paste()` to combine text with a value from a variable such as `paste("My name is", name)`. Doing this successfully will modify your answer to (d), which is fine.

```r
get_mass_from_length_by_name <- function(length, name){
  if (name == "Stegosauria"){
    if (length > 8){
      a = 10.95
      b = 2.64
    } else {
```

```
      a = 8.5
      b = 2.8
    }
  }
  else if (name == "Theropoda"){
    a = 0.73
    b = 3.63
  }
  else if (name == "Sauropoda"){
    a = 214.44
    b = 1.46
  }
  else {
    return(warning(paste("No known estimation for", name)))
  }
  get_mass_from_length(a, b, length)
}

get_mass_from_length_by_name(13, 'Ankylosauria')
```

```
## Warning in get_mass_from_length_by_name(13, "Ankylosauria"): No known
## estimation for Ankylosauria
```

Challenge 2 (optional): Change your function so that it uses two different values of `a` and `b` for *Stegosauria*. When *Stegosauria* is greater than 8 meters long use the equation above. When it is less than 8 meters long, use `a = 8.5` and `b = 2.8`. Run the function for a *Stegosauria* that is 6 meters long.

```
get_mass_from_length_by_name(6, "Stegosauria")
```

```
## [1] 1283.047
```

5. Using `dplyr` Choice Functions (20 points)

For this question, we will be using functions from `dplyr` and data from the `lterdatasampler` package, so make sure you load both of those packages with the `library()` function.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lterdatasampler)
```

a. First, create an object called `crab_size` and assign the value 17 to it.

   Write an `if` statement using the `if_else` function from `dplyr` that says that a crab that is larger than or equal to 20 is "large" and a crab that is smaller than 20 is "small."

b. Make sure that your `crab_size` object is set to 17 (if you overwrote it in (a)).

This time, write a set of nested `if_else` functions that say that if a crab is greater than or equal to 20, it is "large;" if greater than or equal to 10 but less than 20, it is "medium;" and if it is smaller than 10, it is "small."

c. Use the `case_when` function to perform the same task as (b). Again, set the `crab_size` object to 17, as needed.

d. Take a look at the `pie_crab` dataset from the `lterdatasampler` package by running `head(pie_crab)`. We want to create a new column based on the `size` column. Using the `if_else` function inside of a `mutate` function, create a new column called `size_category` based on the same conditions as (a).

   I've placed my new column next to the size column for the sake of the answer key; you don't need to do this.

e. Now use the `case_when` function inside a `mutate` function to create a new column called `size_category3` in the `pie_crab` dataframe that meets the same conditions as (b) and (c).

   I've placed my new column next to the size column for the sake of the answer key; you don't need to do this.

```
print("5a")
```

```
## [1] "5a"
```

```
crab_size <- 17

if_else(crab_size >= 20, "large", "small")
```

```
## [1] "small"
```

```
print("5b")
```

```
## [1] "5b"
```

```
if_else(crab_size > 20, "large",
        if_else(crab_size > 10, "medium", "small"))
```

```
## [1] "medium"
```

```
print("5c")
```

```
## [1] "5c"
```

```
case_when(crab_size > 20 ~ "large",
          crab_size > 10 ~ "medium",
          TRUE ~ "small")
```

```
## [1] "medium"
```

```
print("5d")
```

```
## [1] "5d"
```

```
pie_crab %>%
  mutate(size_category = if_else(size > 20, "large", "small"), .after = size)
```

```
## # A tibble: 392 x 10
##    date       latitude site   size size_category air_temp air_temp_sd water_temp
##    <date>        <dbl> <chr> <dbl> <chr>             <dbl>       <dbl>      <dbl>
##  1 2016-07-24       30 GTM    12.4 small              21.8        6.39       24.5
##  2 2016-07-24       30 GTM    14.2 small              21.8        6.39       24.5
##  3 2016-07-24       30 GTM    14.5 small              21.8        6.39       24.5
##  4 2016-07-24       30 GTM    12.9 small              21.8        6.39       24.5
```

```
##  5 2016-07-24       30 GTM    12.4 small             21.8        6.39       24.5
##  6 2016-07-24       30 GTM    13.0 small             21.8        6.39       24.5
##  7 2016-07-24       30 GTM    10.3 small             21.8        6.39       24.5
##  8 2016-07-24       30 GTM    11.2 small             21.8        6.39       24.5
##  9 2016-07-24       30 GTM    12.7 small             21.8        6.39       24.5
## 10 2016-07-24       30 GTM    14.6 small             21.8        6.39       24.5
## # i 382 more rows
## # i 2 more variables: water_temp_sd <dbl>, name <chr>
```

```
print("5e")
```

```
## [1] "5e"
```

```
pie_crab %>%
  mutate(size_category3 = case_when(size > 20 ~ "large",
                                    size > 10 ~ "medium",
                                    TRUE ~ "small"), .after = size)
```

```
## # A tibble: 392 x 10
##    date       latitude site   size size_category3 air_temp air_temp_sd
##    <date>        <dbl> <chr> <dbl> <chr>              <dbl>       <dbl>
##  1 2016-07-24       30 GTM    12.4 medium              21.8        6.39
##  2 2016-07-24       30 GTM    14.2 medium              21.8        6.39
##  3 2016-07-24       30 GTM    14.5 medium              21.8        6.39
##  4 2016-07-24       30 GTM    12.9 medium              21.8        6.39
##  5 2016-07-24       30 GTM    12.4 medium              21.8        6.39
##  6 2016-07-24       30 GTM    13.0 medium              21.8        6.39
##  7 2016-07-24       30 GTM    10.3 medium              21.8        6.39
##  8 2016-07-24       30 GTM    11.2 medium              21.8        6.39
##  9 2016-07-24       30 GTM    12.7 medium              21.8        6.39
## 10 2016-07-24       30 GTM    14.6 medium              21.8        6.39
## # i 382 more rows
## # i 3 more variables: water_temp <dbl>, water_temp_sd <dbl>, name <chr>
```