Data Science 2

Introduction to advanced data science - spring 2023

February 10, 2023

4_switch_exercise.org

HOW TO `switch` TO `if` (10 min)

Write an explicit *stacked* set of `if` statements that does the same thing as the `integer` version of this `switch` function. Test it with `mynum <- 3` and `mynum <- 0`.

```
foo <- switch(EXPR=mynum,12,34,56,78,NA)
```

Solution:

```
if (mynum == 1) {
  foo <- 12
  paste("Value is", foo )
} else if (mynum == 2) {
  foo <- 34
  paste("Value is", foo)
} else if (mynum == 3) {
  foo <- 56
  paste("Value is", foo)
} else if (mynum == 4) {
  foo <- 78
  paste("Value is", foo)
} else {
  foo <- NA
  paste("Value is", foo)
}
```

```
[1] "Value is NA"
```

Test:

```
mynum <- 3
if (mynum == 1) {
  foo <- 12
  paste("Value is", foo )
} else if (mynum == 2) {
  foo <- 34
  paste("Value is", foo)
} else if (mynum == 3) {
  foo <- 56
  paste("Value is", foo)
} else if (mynum == 4) {
  foo <- 78
  paste("Value is", foo)
} else {
  foo <- NA
  paste("Value is", foo)
}
mynum <- 0
if (mynum == 1) {
  foo <- 12
  paste("Value is", foo )
} else if (mynum == 2) {
  foo <- 34
  paste("Value is", foo)
} else if (mynum == 3) {
  foo <- 56
  paste("Value is", foo)
} else if (mynum == 4) {
  foo <- 78
  paste("Value is", foo)
} else {
  foo <- NA
  paste("Value is", foo)
}
```

```
[1] "Value is 56"
[1] "Value is NA"
```

HOW TO NEST COMPLEX `if` STATEMENTS (30 min)

Since the text of this exercise is quite long, you want to either open it in GitHub next to your editor, or in a 2nd editor window (open it with `C-x 5 2`). Also: please help each other!

Proceed like this: (1) read the **backstory**; (2) look at the variables in the **test suite** below; (3) look at the **tips**; (4) implement the **rules** in a code block named adjust_dosage; (5) run the **test suite**.

Backstory: Suppose you are tasked with computing the precise dosage amounts of a certain drug in a collection of hypothetical experiments. These amounts depend upon some predetermined set of "dosage thresholds" - `numeric` variables `lowdose`, `meddose`, and `highdose` - as well as a predetermined dose level `factor` vector called `doselevel`.

Look at the items (1.-4.) below to see the indended form of these objects. Then **write** a set of *nested* `if` statements that produce a new `numeric` vector called `dosage`, according to the following rules:

IF ELSE Rules:

First, *if* there are any instances of `"High"` in `doselevel`, perform the following operations:

- Check *if* `lowdose` is greater than or equal to `10`. If so, overwrite `lowdose` with `10`; *otherwise*, overwrite `lowdose` by itself divided by 2.
- Check *if* `meddose` is greater than or equal to `26`. If so, overwrite `meddose` by `26`.
- Check *if* `highdose` is less than `60`. If so, overwrite `highdose` with `60`, *otherwise* `highdose` by itself multiplied by `1.5`.
- Create a vector named `dosage` with the values of `lowdose` repeated (`rep`) to match the `length` of `doselevel`.
- Overwrite the elements in `dosage` corresponding to the index positions of instances of `"Med"` in `doselevel` by `meddose`.
- Overwrite the elements in `dosage` corresponding to the index positions of instances of `"High"` in `doselevel` by `highdose`.

Otherwise, (i.e. if there are *no* instances of `"High"` in `doselevel`) perform the following operations:

1. Create a new version of `doselevel`, a `factor` vector with levels `"Low"` and `"Med"` only, and label these with `"Small"` and `"Large"` respectively. Check `args(factor)` or `help(factor)` for `labels`.

2. Check to see *if* `lowdose` is less than 15 AND `meddose` is less than `35`. If so, overwrite `lowdose` by itself multiplied by `2` and overwrite `meddose` by itself plus `highdose`.

3. Create a vector named `dosage`, which is the value of `lowdose` repeated (`rep`) to match the `length` of `doselevel`.

4. Overwrite the elements in `dosage` corresponding to the index positions of instances of `"Large"` in `doselevel` by `meddose`.

Tip 1: In the code block below, `foo` is a `factor` vector with the `levels` `"up"` and `"down"`, and the `labels` `"one"` and `"two"`:

```
foo <- factor(c("down","up","up","down","up"),
              levels=c("up","down"),
              labels=c("one","two"))
foo
```

```
[1] two one one two one
Levels: one two
```

Tip 2: In the code block below, `bar` is a `vector` of numbers with the same `length` as `foo`, and we extract the values of `bar` corresponding to the index positions of instances of `"up"` (labeled `"one"`) in `foo` and overwrite them with `0`.

```
bar <- rep(10,length(foo))  # define bar
foo=="one"  # must use the label for the levels
bar[foo=="one"]
bar[foo=="one"] <- 0
bar
```

```
[1] FALSE  TRUE  TRUE FALSE  TRUE
[1] 10 10 10
[1] 10  0  0 10  0
```

TEST SUITE:

1. Starting dose values and `dosage` after running the solution code

   ```
   lowdose <- 12.5
   meddose <- 25.3
   highdose <- 58.1
   ```

```
doselevel <- factor(
  x = c("Low","High","High","High","Low","Med","Med"),
  levels=c("Low","Med","High"))
if (any(doselevel=="High")) {
  if (lowdose >= 10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose >= 26) {
    meddose <- 26
  }
  if (highdose < 60) {
    highdose <- 60
  } else {
    highdose <- highdose * 1.5
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <- factor(doselevel,
                      levels=c("Low","Med"),
                      labels=c("Small","Large"))
  if (lowdose < 15 && meddose < 35) {
    lowdose <- lowdose * 2
    meddose <- meddose + highdose
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
dosage

[1] 10.0 60.0 60.0 60.0 10.0 25.3 25.3
```

2. Starting values, `dosage` and `doselevel` after running the solution code:

```
lowdose <- 12.5
meddose <- 25.3
highdose <- 58.1
```

```
doselevel <- factor(
  x = c("Low","Low","Low","Med","Low","Med","Med"),
  levels=c("Low","Med","High"))
if (any(doselevel=="High")) {
  if (lowdose >= 10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose >= 26) {
    meddose <- 26
  }
  if (highdose < 60) {
    highdose <- 60
  } else {
    highdose <- highdose * 1.5
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <- factor(doselevel,
                      levels=c("Low","Med"),
                      labels=c("Small","Large"))
  if (lowdose < 15 && meddose < 35) {
    lowdose <- lowdose * 2
    meddose <- meddose + highdose
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
dosage
doselevel

[1] 25.0 25.0 25.0 83.4 25.0 83.4 83.4
[1] Small Small Small Large Small Large Large
Levels: Small Large
```

3. Starting values, `dosage` and `doselevel` after running the solution code:

```
lowdose <- 9
meddose <- 49
highdose <- 61
doselevel <- factor(
  x = c("Low","Med","Med"),
  levels=c("Low","Med","High"))
if (any(doselevel=="High")) {
  if (lowdose >= 10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose >= 26) {
    meddose <- 26
  }
  if (highdose < 60) {
    highdose <- 60
  } else {
    highdose <- highdose * 1.5
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <- factor(doselevel,
                      levels=c("Low","Med"),
                      labels=c("Small","Large"))
  if (lowdose < 15 && meddose < 35) {
    lowdose <- lowdose * 2
    meddose <- meddose + highdose
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
dosage
doselevel
```

```
[1]  9 49 49
[1] Small Large Large
Levels: Small Large
```

4. Starting values, `dosage` and `doselevel` after running the solution code:

```
lowdose <- 9
meddose <- 49
highdose <- 61
doselevel <- factor(
  x = c("Low","High","High","High","Low","Med","Med"),
  levels=c("Low","Med","High"))
if (any(doselevel=="High")) {
  if (lowdose >= 10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose >= 26) {
    meddose <- 26
  }
  if (highdose < 60) {
    highdose <- 60
  } else {
    highdose <- highdose * 1.5
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <- factor(doselevel,
                      levels=c("Low","Med"),
                      labels=c("Small","Large"))
  if (lowdose < 15 && meddose < 35) {
    lowdose <- lowdose * 2
    meddose <- meddose + highdose
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
dosage
```

```
[1]  4.5 91.5 91.5 91.5  4.5 26.0 26.0
```

Solution:

```
if (any(doselevel=="High")) {
  if (lowdose >= 10) {
    lowdose <- 10
  } else {
    lowdose <- lowdose/2
  }
  if (meddose >= 26) {
    meddose <- 26
  }
  if (highdose < 60) {
    highdose <- 60
  } else {
    highdose <- highdose * 1.5
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Med"] <- meddose
  dosage[doselevel=="High"] <- highdose
} else {
  doselevel <- factor(doselevel,
                      levels=c("Low","Med"),
                      labels=c("Small","Large"))
  if (lowdose < 15 && meddose < 35) {
    lowdose <- lowdose * 2
    meddose <- meddose + highdose
  }
  dosage <- rep(x = lowdose, length=length(doselevel))
  dosage[doselevel=="Large"] <- meddose
}
```

HOW TO USE `switch` with `ifelse` (5 min)

Assume the object `mynum` will only ever be a single integer between 0 and 9. Use `ifelse` and `switch` to produce a command that takes in `mynum` and returns a matching `character` string for all possible values 0,1,...,9:

- Supplied with 3, for example, it should return `"three"`.
- Supplied with 0, it should return `"zero"`.

Solution:

```
ifelse(test = (mynum>0),
       yes = switch(mynum,
                    "one",
                    "two",
                    "three",
                    "four",
                    "five",
                    "six",
                    "seven",
                    "eight",
                    "nine"),
       no = "zero")
```

```
[1] "zero"
```

Test the solution:

```
mynum <- 3
ifelse(test = (mynum>0),
       yes = switch(mynum,
                    "one",
                    "two",
                    "three",
                    "four",
                    "five",
                    "six",
                    "seven",
                    "eight",
                    "nine"),
       no = "zero")
mynum <- 0
ifelse(test = (mynum>0),
       yes = switch(mynum,
                    "one",
                    "two",
                    "three",
                    "four",
                    "five",
                    "six",
                    "seven",
                    "eight",
```

```
                    "nine"),
       no = "zero")
```

```
[1] "three"
[1] "zero"
```