Transform the percentage cover values into ordinal scale - Solution

David Zelený

Solution R file should contain the definition of the `transform2brbl` function, and testing it on value sequence and `vltava` data, including the output of those tests.

Regarding the function, the first option is to use a sequence of `ifelse (CONDITION, EXPR1, EXPR2)` functions, which can take as an argument vector or matrix:

```
transform2brbl_1 <- function (data)
  {
  ifelse (data == 0, 0,
    ifelse (data <= 1, 1,
      ifelse (data <= 2, 2,
        ifelse (data <= 3, 3,
          ifelse (data <= 25, 4,
            ifelse (data <= 50, 5,
              ifelse (data <= 75, 6, 7)))))))
  }
```

An alternative option is to use the `for` loop and evaluate each element separately using nested sequence of `if (CONDITION) EXPR1 else EXPR2`:

```
transform2brbl_2 <- function (data)
{
  t.data <- as.matrix(data)
  for (ro in seq (1, nrow (t.data)))
    for (co in seq (1, ncol (t.data)))
    {
      t.data[ro,co] <- if (t.data[ro,co] == 0) 0 else
        if (t.data[ro,co] <= 1) 1 else
          if (t.data[ro,co] <= 2) 2 else
            if (t.data[ro,co] <= 3) 3 else
              if (t.data[ro,co] <= 25) 4 else
                if (t.data[ro,co] <= 50) 5 else
                  if (t.data[ro,co] <= 75) 6 else 7
    }
  dim (t.data) <- dim (data)
  dimnames (t.data) <- dimnames (data)
  return (t.data)
}
```

The function needs to cope with two data formats, vector and data.frame (`1:100` is a vector, while `vltava.spe` is a data.frame). At the same time, since `if else` statement can evaluate only a single CONDITION, it has to loop through the two-dimensional data frame element by element (row by row and column by column, see the two `for` loops). Eventually, I need to make sure that if the original `data` were a vector, the output will also become vector; this is achieved by assigning the `t.data` the same `dim` (dimension) as was the original `data`. If this was a vector, the `dim` function returns `NULL`, and assigning this to the matrix will turn it into the vector; if it was originally matrix or data.frame, the number of rows and columns will remain. I also need to make sure to assign dimension names in some way, here using the function `dimnames`.

I can also use this solution and extend it, every time make sure that the value really fits the specific range (e.g. it is larger than 3 but smaller or equal to 25). This is not really necessary, since the previous solution is already precise enough, but it is possible to do:

```
transform2brbl_3 <- function (data)
{
  t.data <- as.matrix (data)
  for (ro in seq (1, nrow (t.data)))
    for (co in seq (1, ncol (t.data)))
    {
      t.data[ro,co] <- if (t.data[ro,co] == 0) 0 else
        if (t.data[ro,co] > 0 & t.data[ro,co] <= 1) 1 else
          if (t.data[ro,co] > 1 & t.data[ro,co] <= 2) 2 else
            if (t.data[ro,co] > 2 & t.data[ro,co] <= 3) 3 else
              if (t.data[ro,co] > 3 & t.data[ro,co] <= 25) 4 else
                if (t.data[ro,co] > 25 & t.data[ro,co] <= 50) 5 else
                  if (t.data[ro,co] > 50 & t.data[ro,co] <= 75) 6 else 7
    }
  dim (t.data) <- dim (data)
  dimnames (t.data) <- dimnames (data)
  return (t.data)
}
```

In a final variation on `for` loop and `if else` sequence is to replace `if (CONDITION) EXPR1 else EXPR2` structure by only `if (CONDITION) EXPR`. This makes the script more complicated, since I have to use a double comparison (e.g. `x > 3 & x <= 25`), because each line of script with `if` statement is evaluated independently from the others:

```
transform2brbl_4 <- function (data)
{
  t.data <- as.matrix (data)
  for (ro in seq (1, nrow (t.data)))
    for (co in seq (1, ncol (t.data)))
    {
      if (t.data[ro,co] == 0) t.data[ro,co] <- 0
      if (t.data[ro,co] > 0 & t.data[ro,co] <= 1) t.data[ro,co] <- 1
      if (t.data[ro,co] > 1 & t.data[ro,co] <= 2) t.data[ro,co] <- 2
      if (t.data[ro,co] > 2 & t.data[ro,co] <= 3) t.data[ro,co] <- 3
      if (t.data[ro,co] > 3 & t.data[ro,co] <= 25) t.data[ro,co] <- 4
      if (t.data[ro,co] > 25 & t.data[ro,co] <= 50) t.data[ro,co] <- 5
      if (t.data[ro,co] > 50 & t.data[ro,co] <= 75) t.data[ro,co] <- 6
      if (t.data[ro,co] > 75 & t.data[ro,co] <= 100) t.data[ro,co] <- 7
    }
  dim (t.data) <- dim (data)
  dimnames (t.data) <- dimnames (data)
  return (t.data)
}
```

A completely different solution, introduced by Lu Chung-Hsuan (a student in REcol), is to use function `replace` (check `?replace`). The function `replace` has three arguments: `x`, which loads an object (vector, matrix, or data.frame), `list`, which needs to be an index vector or logical vector, and finally `values`, which contains replacement values.

```
transform2brbl_5 <- function (data)
  {
  data1 <- (data > 0 & data <= 1)
  #Save the data points into the same type of framework as input data as T/F values
  data1. <- replace (data1, data1 > 0, 1)
  #Replace the 'TRUE' into a suitable value depending on the ordinal-percentage relationship
  data2 <- (data > 1 & data <= 2)
  data2. <- replace (data2, data2 > 0, 2)

  data3 <- (data > 2 & data <= 3)
  data3. <- replace (data3, data3 > 0, 3)

  data4 <- (data > 3 & data <= 25)
  data4. <- replace (data4, data4 > 0, 4)

  data5 <- (data > 25 & data <= 50)
  data5. <- replace (data5, data5 > 0, 5)

  data6 <- (data > 50 & data <= 75)
  data6. <- replace (data6, data6 > 0, 6)

  data7 <- (data > 75 & data <= 100)
  data7. <- replace (data7, data7 > 0, 7)

  datatotal <- (data1. + data2. + data3. + data4. + data5. + data6. + data7.)
  #Simply summing up to merge every replaced data
  return (datatotal)
  }
```