

control-structures

@MaxMLang (Github)

find_opposite

Finds the ‘opposite’ of a value in a two column dataframe.

Inputs

- database: A two column dataframe (colnames(value, opposite)), where each row represents a pair
- value: A value in the dataframe you want to find the opposite of.

Output

- The respective opposite value

```
find_opposite <- function(database, value) {  
  if (value %in% database$value) {  
    position <- which(database$value == value)[[1]]  
    database$opposite[[position]]  
  } else {  
    position <- which(database$opposite == value)[[1]]  
    database$value[[position]]  
  }  
}
```

Worked example

```
data <- data.frame(list(value = c("positive", "white", "Ying", "left"),  
                        opposite = c("negative", "black", "Yang", "right")))
```

```
find_opposite(data, "positive")
```

```
## [1] "negative"
```

```
find_opposite(data, "Ying")
```

```
## [1] "Yang"
```

```
# works both ways
```

```
find_opposite(data, "right")
```

```
## [1] "left"
```

cellular_auto

A “cellular automaton” is a discrete model of state evolution. We consider a state of a vector with “cells” of values 0 or 1, for example the vector `c(0, 1, 0, 1, 0, 0, 0)`. The state now changes according to a special

rule: Every cell changes to state 1 whenever its immediate left neighbour has state 1. A cell that has state 1 remains at state 1. Each cell that has no left neighbour or a left neighbour that is 0, and that is also 0 itself, remains 0 otherwise.

Inputs

- start.state: A vector only containing 0 and 1 representing the initial state of the cell.

Output

A list of n process steps until the process is completed.

Code

```
cellular_auto <- function(start.state) {  
  results <- list(start.state)  
  state <- start.state  
  repeat {  
    # shift the state to the right by 1  
    state.shifted <- c(0, head(state, -1))  
    # every state where state or state.shifted is 1 should be 1, others 0  
    state.next <- ifelse(state == 1 | state.shifted == 1, 1, 0)  
    # if nothing changed, we are done  
    if (all(state.next == state)) break  
    results[[length(results) + 1]] <- state <- state.next  
  }  
  results  
}
```

Worked example

```
startcell <- c(0,0,1,0,1,0,0,1,0)  
cellular_auto(startcell)
```

```
## [[1]]  
## [1] 0 0 1 0 1 0 0 1 0  
##  
## [[2]]  
## [1] 0 0 1 1 1 1 0 1 1  
##  
## [[3]]  
## [1] 0 0 1 1 1 1 1 1 1
```