5b_prog_control_flow_lec4

Code

Control flow

1. if (exactly same with C)

1. Syntax

```
if (condition) {
    commands when TRUE # do if TRUE
} else {
    commands when FALSE # do if FALSE
}
```

2. notes:
    1. if only 1 command, can omit `{}`
    2. `else` part is optional
3. vectorized version if-else
    1. syntax

    ```
    ifelse(test, yes_expr, no_expr)
    ```

    2. eg if x is a vector, can compare each element and do things

    ```
    > x <- -5:5
    > ifelse(x<0, x, -x)
     [1] -5 -4 -3 -2 -1  0 -1 -2 -3 -4 -5
    ```

    3. Note
        1. if meet NA in an element of a vector, won't have error and will result in NA

2. loop

2.1. loop special syntax

1. `break`, `next` (`next` causes control to return immediately to the top of the loop)

2.2. for loop

1. syntax:

```
for (x in v) { commands }
```

2. notes:
    1. `v` is usually a vector but also could be a list
    2. if only 1 command, can omit `{}`
    3. Like python, the x is passed by value, modifying x won't change v
    4. If change v in the loop:
       v in the for loop is evaluated at the start of the loop, changing it subsequently does not affect the loop.
    5. The variables defined in the loop body can still be used when the loop ends

```
my.ran2 <- function(n, dist="norm"){
    # default value of dist is "norm"
    # version 2: using switch
    switch(dist, "norm"=rnorm(n), "uniform"=runif(n),
    stop("Unknown distribution"))
}
```

3. notes:
    1. An expression type with character string always matched to the listed cases.
    2. An expression which is not a character string then this exp is coerced to integer.
    3. For multiple matches, the first match element will be used.
    4. No default argument case is available there in R switch case.
    5. An unnamed case can be used, if there is no matched case (see eg2).

2.3. while loop (exactly same with C)

1. syntax:

```
while (condition) {statements}
```

2. notes:

3. repeat loop

1. syntax:

```
repeat { statements
  ...
  if (condition) break
}
```

2. notes: 1.Idea: loop forever, only until receice break

3. switch

1. Syntax

```
switch(expression, case1, case2, case3....)
```

2. eg
    1. basic

    ```
    # chars
    > switch('b','a'="red",'b'="green",'c'="blue")
    [1] "green"
    > switch(' a' ,a="red",b="green",c="blue")    # without single quote also can
    [1] "red"

    # integer
    switch(4, "Geeks1", "Geeks2", "Geeks3", "Geeks4", "Geeks5", "Geeks6")
    [1] "Geeks4"
    ```

    2. string as match