Sane's STATS Academy of Statistics

Control Structures

Sane's STATS Academy of Statistics

Control Structures

- if, else
- for loop
- while loop
- break – breaking an execution of a loop
- next – skipping an iteration

if , else Structure

- Conditional processing: If the condition given in the if() is true then the corresponding code block gets executed otherwise else code block is executed

if Syntax : if(condition) {

statements

}

If-else syntax : if(condition) {

statements

} else {

statements

}

```
a <- 34000
b <- 50000
if(a+b>10000) {
  paste("Total greater than 10000")
} else {
  paste("Total not greater than 10000")
}
```

for loop

- Loop can be created with for() using following syntax :

for(var in seq) expr

```
> for(i in 1:4) {
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
```

while( ) loop

- Loop can be generated with while() using following syntax :

while(cond) expr

- So long as the condition remains true the body of loop continues to execute

```
> cnt <- 1
> while(cnt < 5) {
+   print(cnt)
+   cnt <- cnt + 1
+ }
[1] 1
[1] 2
[1] 3
[1] 4
```

Breaking an execution of loop

- Breaking a loop can be possible with break statement

```
> for(i in 1:4) {
+   if(i==3) break
+   print(i)
+ }
[1] 1
[1] 2
```

Skipping an iteration of a loop

- For skipping the iteration of the loop, next statement is used

```
> for(i in 1:4) {
+   if(i==3) next
+   print(i)
+ }
[1] 1
[1] 2
[1] 4
```

Sane's STATS Academy of Statistics

Functions

Sane's STATS Academy of Statistics

Some functions we will cover…

- str
- seq
- table
- prop.table
- cut
- sample
- log
- exp
- Ifelse
- attach

The above functions are used quiet frequently while preparing data

str()

- str function displays the internal structure of an R object
- It can be called as a diagnostic function, we often use to know about the object before we work on it

```
> str(items)
'data.frame':   25 obs. of  6 variables:
 $ Item.ID  : Factor w/ 25 levels "121 001","121 002",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Item.Name: Factor w/ 25 levels "Artline EK-999XF Metallic Ink Marker - Silver",..: 11 18 10 17 20 16 25
 12 19 7 ...
 $ Item.Type: Factor w/ 4 levels "Highlighter",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Brand    : Factor w/ 12 levels "Artline","Camlin",..: 6 8 6 8 10 6 12 6 9 3 ...
 $ Price    : int  69 135 125 135 60 92 160 316 179 90 ...
 $ UOM      : Factor w/ 2 levels "Pack","Piece": 2 1 2 1 1 2 1 1 2 1 ...
```

seq()

- For sequence generation, seq() is used

Syntax :

seq(from = 1, to = 1, by = incr/decr...)

```
> seq(1,20)
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
> seq(1,20,by=4)
[1]  1  5  9 13 17
```

table()

- Frequency table and Cross-tabulation can be generated with table()

Syntax: table(var1,var2,…)

```
> table(survey$Exer)

Freq None Some
 115   24   98
```

```
> table(survey$Sex)

Female   Male
   118    118
> table(survey$Sex, useNA = "ifany")

Female   Male   <NA>
   118    118      1
```

table()

- Multivariate Frequencies

```
> table(survey$Sex,survey$Exer, useNA = "ifany")

        Freq None Some
Female    49   11   58
Male      65   13   40
<NA>       1    0    0
```

prop.table()

- The function prop.table() computes the proportions
- We can convert those proportions into percentages by multiplying them by 100

Syntax: prop.table(table)
prop.table(table,1) # row proportions
prop.table(table,2) # column proportions

```
> prop.table(table(items$Item.Type))

Highlighter      Marker         Pen      Refill
       0.12        0.20        0.64        0.04
```

```
> prop.table(table(items$Item.Type,items$Brand))

             Artline Camlin Cello Lamy Luxor Parker Pierre Cardin Pilot Puro Reynolds Sheaffer Staedtler
Highlighter     0.00   0.08  0.00 0.00  0.00   0.00           0.00  0.04 0.00     0.00     0.00     0.00
Marker          0.08   0.08  0.00 0.00  0.04   0.00           0.00  0.00 0.00     0.00     0.00     0.00
Pen             0.00   0.00  0.04 0.00  0.00   0.16           0.08  0.08 0.04     0.08     0.04     0.12
Refill          0.00   0.00  0.00 0.04  0.00   0.00           0.00  0.00 0.00     0.00     0.00     0.00
```

Sane's STATS Academy of Statistics

prop.table() Contd…

```
> prop.table(table(items$Item.Type,items$Brand),1)

                Artline    Camlin     Cello      Lamy     Luxor    Parker Pierre Cardin
  Highlighter 0.0000000 0.6666667 0.0000000 0.0000000 0.0000000 0.0000000      0.0000000
  Marker      0.4000000 0.4000000 0.0000000 0.0000000 0.2000000 0.0000000      0.0000000
  Pen         0.0000000 0.0000000 0.0625000 0.0000000 0.0000000 0.2500000      0.1250000
  Refill      0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000      0.0000000

                 Pilot      Puro  Reynolds  Sheaffer Staedtler
  Highlighter 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000
  Marker      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  Pen         0.1250000 0.0625000 0.1250000 0.0625000 0.1875000
  Refill      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
```

```
> prop.table(table(items$Item.Type,items$Brand),2)

                Artline    Camlin     Cello      Lamy     Luxor    Parker Pierre Cardin
  Highlighter 0.0000000 0.5000000 0.0000000 0.0000000 0.0000000 0.0000000      0.0000000
  Marker      1.0000000 0.5000000 0.0000000 0.0000000 1.0000000 0.0000000      0.0000000
  Pen         0.0000000 0.0000000 1.0000000 0.0000000 0.0000000 1.0000000      1.0000000
  Refill      0.0000000 0.0000000 0.0000000 1.0000000 0.0000000 0.0000000      0.0000000

                 Pilot      Puro  Reynolds  Sheaffer Staedtler
  Highlighter 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000
  Marker      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
  Pen         0.6666667 1.0000000 1.0000000 1.0000000 1.0000000
  Refill      0.0000000 0.0000000 0.0000000 0.0000000 0.0000000
```

Sane's STATS Academy of Statistics

cut( )

Syntax : cut(x, breaks, include.lowest = FALSE,…)

- cut() divides the range of x into intervals and codes the values in x according to which interval they fall.
- The leftmost interval corresponds to level one, the next leftmost to level two and so on.
- breaks : either a numeric vector of two or more unique cut points or a single number (greater than or equal to 2) giving the number of intervals into which x is to be cut.

```
> cut(items$Price,breaks=4)
 [1] (49.6,154] (49.6,154] (49.6,154] (49.6,154] (49.6,154] (49.6,154] (154,258]  (258,361]
 [9] (154,258]  (49.6,154] (258,361]  (258,361]  (154,258]  (258,361]  (49.6,154] (258,361]
[17] (49.6,154] (49.6,154] (154,258]  (49.6,154] (258,361]  (49.6,154] (361,465]  (154,258]
[25] (49.6,154]
Levels: (49.6,154] (154,258] (258,361] (361,465]
```

```
> table(cut(items$Price,breaks=4))

(49.6,154]  (154,258]  (258,361]  (361,465]
        13          5          6          1
```

Sane's STATS Academy of Statistics

cut( ) Contd…

```
> cut(items$Price,breaks=c(40,50,65,80,100,300,400,500))
 [1] (65,80]   (100,300] (100,300] (100,300] (50,65]   (80,100]  (100,300] (300,400] (100,300]
[10] (80,100]  (300,400] (100,300] (100,300] (300,400] (100,300] (100,300] (40,50]   (80,100]
[19] (100,300] (80,100]  (100,300] (80,100]  (400,500] (100,300] (100,300]
Levels: (40,50] (50,65] (65,80] (80,100] (100,300] (300,400] (400,500]
```

sample()

- Sample function gives a random sample of specified size
- Syntax : sample(x, size, replace = FALSE)
  - x : Either a vector of one or more elements from which to choose, or a positive integer
  - n : a positive number, the number of items to choose from
  - replace : Should sampling be with replacement?

Sane's STATS Academy of Statistics

sample() Contd…

```
> sample(items$Price, size=8)
[1] 190 320 465 100 316 270 300  92
```

```
> sample(items$Price, size=8,replace = TRUE)
[1] 225 125  69  69  60 135  90 270
```

Sane's STATS Academy of Statistics

log()

- For calculating the logarithm, we can use log() function

Syntax:

log(x)

log10(x)

log2(x)

log1p(x)

log() Contd…

- log computes logarithms, by default natural logarithms,
- log10 computes common (i.e., base 10) logarithms,
- log2 computes binary (i.e., base 2) logarithms.
- The general form log(x, base) computes logarithms with base base.
- log1p(x) computes *log(1+x)* accurately also for $|x| << 1$

Sane's STATS Academy of Statistics

log() and exp()

```
> log(2)
[1] 0.6931472
> log(2,10)
[1] 0.30103
> log1p(2)
[1] 1.098612
> log10(2)
[1] 0.30103
> log2(2)
[1] 1
```

```
> exp(0.6931472)
[1] 2
> 10^0.30103
[1] 2
> expm1(1.098612)
[1] 1.999999
> 10^0.30103
[1] 2
> 2^1
[1] 2
```

exp()

- exp computes the exponential function.
- expm1(x) computes exp(x) - 1 accurately also for |x| << 1.

Syntax : exp(x)
expm1(x)

```
> exp(0.6931472)
[1] 2
> expm1(0.6931472)
[1] 1
```

Sane's STATS Academy of Statistics

ifelse()

- In a simple way, we can treat ifelse() as a functional version of the if-else structure

Syntax : ifelse(condition, *true-value*, *false-value*)

- If the condition is TRUE then the function returns *true-value* otherwise it returns *false-value*

```
> v <- c(23,13,9,24,09,3,14,8,18,20)
> result <- ifelse(v > 10, "Pass","Fail")
> result
 [1] "Pass" "Pass" "Fail" "Pass" "Fail" "Fail" "Pass" "Fail"
 [9] "Pass" "Pass"
```

Mean and Variance Functions

- mean()
- sd()
- var()
- Each of the functions above have the syntax usage in the following way: function-name(variable-name,na.rm)
  - na.rm by default is FALSE. It should be set to TRUE if we want to ignore the NA values while computing

```
> mean(items$Price,na.rm = TRUE)
[1] 180.4
> sd(items$Price,na.rm = TRUE)
[1] 105.7107
> var(items$Price,na.rm = TRUE)
[1] 11174.75
```

summary()

- For numerical variables, summary function outputs the Minimum, Maximum, 1st Quartile, Median, 3rd Quartile and Mean

```
> summary(items$Price)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   50.0   100.0   135.0   180.4   270.0   465.0
```

- For categorical variables, summary function outputs the frequency counts

```
> summary(items$Item.Type)
Highlighter      Marker         Pen      Refill
          3           5          16           1
```

summary()

```
> summary(items)
    Item.ID                                                    Item.Name
 121 001: 1   Artline EK-999XF Metallic Ink Marker - Silver          : 1
 121 002: 1   Artline EK157R Whiteboard Marker - Black, Pack of 10 : 1
 121 003: 1   Camlin CD - DVD Marker Pen, Blue - Pack of 10         : 1
 121 004: 1   Camlin Office Highlighter - Pack of 5 Assorted Colors: 1
 121 005: 1   Camlin Office Highlighter Pen, Yellow                 : 1
 121 006: 1   Camlin PB White Board Marker Pen, Blue                : 1
 (Other):19   (Other)                                               :19
      Item.Type             Brand       Price             UOM
 Highlighter: 3   Camlin        :4   Min.    : 50.0   Pack :15
 Marker     : 5   Parker        :4   1st Qu.:100.0   Piece:10
 Pen        :16   Pilot         :3   Median :135.0
 Refill     : 1   Staedtler     :3   Mean   :180.4
                  Artline       :2   3rd Qu.:270.0
                  Pierre Cardin:2   Max.   :465.0
                  (Other)       :7
```

- We have some few more functions in R like predict(), plot() which behave according to the class of the argument

attach()

- The data is attached to the **R** search path with attach().
- Data is searched by **R** when evaluating a variable, so objects in the data can be accessed by simply giving their names.

Syntax : attach(data)

Hence, instead of typing…

```
> table(items$Item.Type)

Highlighter      Marker         Pen      Refill
          3           5          16           1
```

```
> mean(items$Price,na.rm = TRUE)
[1] 180.4
```

It can be simply typed as…

```
> attach(items)
> table(Item.Type)
Item.Type
Highlighter      Marker         Pen      Refill
          3           5          16           1
> mean(Price)
[1] 180.4
```

Sane's STATS Academy of Statistics

Creating Functions

- Some tasks which may be repeated in different situations can be coded as a function
- A function has inputs and outputs
- Functions play a very important role in interactive graphics technologies like Tibco Spotfire and Shiny
- We create user defined functions by the following syntax:

```
Function-Name <- function(argument-list) {
                                    statements
                                }
```

Sane's STATS Academy of Statistics

Function Examples

```
add <- function(a,b,c){
  a+b+c
}

# OR

add <- function(a,b,c){
  return(a+b+c)
}
```

```
> descriptive <- function(input) {
+   df <- data.frame(Mean = mean(input,na.rm = TRUE),SD = sd(input,na.rm = TRUE))
+   df
+ }
```

Calling the function:

```
> add(23,24,12)
[1] 59
```

```
> descriptive(items$Price)
   Mean        SD
1 180.4 105.7107
```