

## **SIMULATION WITH S-PLUS:**

### **1- c Combine Values into a Vector or List**

#### **DESCRIPTION**

Concatenates objects into a vector or a list. Generally the result is a list if one or more of the objects is a list, and a vector otherwise.

#### **USAGE**

`c(..., recursive=F)`

#### **REQUIRED ARGUMENTS**

...any S-PLUS objects. Missing values (NAs) are allowed. The name of arguments of length 1 become part of the names of the resulting vector.

#### **OPTIONAL ARGUMENTS**

`recursive=` logical flag: should the combining be done recursively?

#### **VALUE**

vector or list which is the combination of all values from all arguments to the function. If `recursive` is TRUE, arguments with recursive modes are effectively unlisted (see `unlist`) before they are used. The mode of the result is the most general of all the modes in the arguments (or the unlisted arguments, if `recursive=TRUE`). In particular, list objects can be combined this way. The names attribute of the result will be generated from the argument names, if any, plus the names of the combined objects. See `unlist` for the rule used. Attributes of objects that are bound with other objects are deleted.

#### **DETAILS**

Arguments that are NULL or length 0 do not contribute elements to the result.

Note that `c(...,recursive=r)` is equivalent to `unlist(list(...), recursive=r)`.

This is a generic function; currently there are no methods for it.

## EXAMPLES

```
c(1:10, 1:5, 1:10)

c(2, 3, 5, 7, 11, 13)

c(a=1,b=2) # vector with names "a" and "b"

c(states, "Washington DC")

# a list of 4 numeric vectors

c(list(1:3, a=3:6), list(8:23, c(3, 8, 39)))

# a numeric vector of length 26

c(list(1:3, a=3:6), list(8:23, c(3, 8, 39)), recursive=T)

# build x, element by element

# useful if final length not known in advance

x <- numeric(0)

for(i in possibles)

  if(test(i)) x <- c(x, fun(i))
```

## 2- seq Sequences of Numbers

### DESCRIPTION

Creates a vector of evenly spaced numbers. The beginning, end, spacing and length of the sequence can be specified.

This function is generic (see Methods); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: dates.

### USAGE

```
seq(.....)
```

```
seq.default(from=<<see below>>, to=<<see below>>, by=<<see below>>,
length=<<see below>>, along=NULL)
```

from:to # as operator

### OPTIONAL ARGUMENTS

**from** starting value of the sequence. If **to**, **by** and **length** are all given, the value for this is inferred; otherwise, the default is 1. This is required in the operator form.

**to** ending value of sequence, a value less than **from** is allowed. If **from**, **by** and **length** are all given, the value for this is inferred; otherwise, the default is 1. This is required in the operator form.

**by** spacing between successive values in the sequence. If **from**, **to** and **length** are all given, the value for this is inferred; otherwise, the default is 1.

**length** number of values in the result. If **from**, **to** and **by** are all given, the value for this is inferred.

**along** an object. The length of the object is used as the length of the returned value.

It is an error to specify all of the first four arguments.

### VALUE

a numeric vector with values (from, from+by, from+2\*by, ... to). **from** may be larger or smaller than **to**. If **by** is specified, it must have the appropriate sign to generate a finite sequence.

### DETAILS

If there is one unnamed argument that is of length 1 and numeric, then a sequence from 1 to the value of the argument is returned with step sizes of 1.

If the difference between **from** and **to** is not a multiple of **by**, then the sequence stops at the last value not past **to**.

To generate a sequence from 1 to  $\text{length}(x)$  to parallel an object **x**, use `seq(along=x)`. This same behavior results from giving `seq` a single unnamed argument that is not numeric. The expression `seq(along=x)` produces the desired result even if the length of **x** is 0 or 1. Except for these two cases, either `seq(x)` or `1:length(x)` will produce the same result.

### NOTE

The `:` operator has a high precedence (see Syntax); for example, to create a sequence from 1 to  $n-1$ , parentheses are needed `1:(n-1)`.

## EXAMPLES

```
seq(5) # 1, 2, 3, 4, 5 # the same thing as 1:5
```

```
5:1 # 5, 4, 3, 2, 1
```

```
seq(-5) # 1, 0, -1, -2, -3, -4, -5
```

```
1.1 : 5 # 1.1, 2.1, 3.1, 4.1
```

```
seq(0, 1, .01) # 0, .01, .02, .03, ..., 1.
```

```
seq(-pi, pi, length=100) # 100 values from -pi to pi
```

## 3- Subscript Extract or Replace Parts of an Object - Generic operator

### DESCRIPTION

Extract parts of a vector, a list, or an array. If to the left of an assignment, then that portion of the object is changed.

This function is generic (see Methods); method functions can be written to handle specific classes of data. Classes which already have methods for this function include: `anova`, `data.frame`, `factor`, `model.matrix`, `smooth`, `tree`.

### USAGE

```
x[i]  
x[i, j, ...]  
x[i, j, ..., drop=T]  
x[[i]]  
x[[i, j, ...]]  
x$name
```

### REQUIRED ARGUMENTS

- x any object.
- i, j subscript expressions, used to identify which elements to extract or replace. The expressions may be empty (meaning all possible subscripts), logical, numeric, or character.
- name a name or quoted string.

### OPTIONAL ARGUMENTS

drop logical flag, should dimensions of length 1 be dropped. For example, assume you have a 5 by 10 matrix  $m$ .  $m[,1]$  will produce a vector of length 5 and  $m[,1,drop=F]$  will produce a 5 by 1 matrix. The  $drop=F$  argument is most useful in functions where consistency is important  $\frac{3}{4}$  an expression  $m[,i,drop=F]$  will always produce a 2-dimensional matrix regardless of the length of  $i$ .

## VALUE

These are the extraction functions, returning some elements or properties of an object.

## SIDE EFFECTS

They may also appear on the left of an assignment operation, to carry out replacement in an object.

## DETAILS

Vector subscripts are generated with  $x[i]$  when  $i$  and  $x$  are both vectors. (Note that an array is also a vector, hence arrays can be subscripted as vectors - both intentionally and unintentionally.) The result of the expression is to extract or replace elements of  $x$  corresponding to a vector of positive indices computed according to the value of  $i$ .

If  $i$  is empty, all of  $x$  is extracted or replaced, without affecting the attributes of  $x$ . If  $i$  is logical the indices are produced by starting at 1 and selecting the numbers for which the corresponding element  $i$  is TRUE. If  $i$  is shorter than  $\text{length}(x)$ , it is extended by cyclic repetition. It can be longer than  $\text{length}(x)$  as well, with no change in the computation of indices. If  $i$  has mode "character", indices are determined by matching the elements of  $i$  against the attribute names( $x$ ). Unmatched names (including the case that there is no such attribute) index outside the current length of  $x$ . If  $i$  is numeric, and  $\text{all}(i \leq 0)$  the indices consist of the elements of  $\text{seq}(\text{along}=x)$  that do not match any elements in  $-i$ . Otherwise,  $i$  itself is taken to be the indices. The indices can have any positive values, 0, or NA. Zeros are dropped before using the indices.

The computed indices are used for extraction or replacement. The rule for extraction is that the value of  $x[i]$  has the same mode as  $x$ , and the same length as the number of indices. The elements of  $x[i]$  are the elements of  $x$  corresponding to the indices, except if the indices are greater than the length of  $x$  or are NA. In either of those exceptions the returned elements are "missing"; that is, NA for an atomic mode ("" for character) and NULL for a non-atomic mode. All the attributes of  $x$  will be discarded in the subset, except for the names attribute. The names attribute of  $x[i]$  will be names( $x$ )[ $i$ ].

If there are  $k$  subscripts and  $x$  is a  $k$ -way array, indices identifying a sub-array of  $x$  are computed from the  $i$ -th subscript with respect to  $(1:\text{dim}(x))[i]$ . Character subscripts in the  $i$ -th expression are used relative to the  $i$ -th element of  $\text{dimnames}(x)$ . If  $x$  is a  $k$ -way array

and the single subscript is a matrix with  $k$  columns, vector subscripts, one element per row of  $i$ , are computed in the same way.

The computations for  $x[[i]]$  are identical to the above except that the expression is required to identify a single element to be extracted or replaced. The value returned for extraction is the same as for  $x[i]$  if  $x$  is atomic (e.g., numeric). If  $x$  is recursive (e.g., a list) the single extracted element is returned, not the list of length 1 that  $x[i]$  would produce.

A special case of the `[[` construct is when  $x$  is recursive (e.g., a list) and  $i$  is a list. The components of  $i$  should all have length 1 and be character or numeric. The statement  $x[[i]]$  is equivalent to:

```
x [[ i[[1]] ]] [[ i[[2]] ]] ... [[ i[[length(i)]] ]]
```

For example,  $x[[list("b", 2, 4)]]$  will pick out the 4th component of the 2nd component of the `b` component of  $x$ .

The expression  $x$name$  is the name component of  $x$ . It is equivalent to  $x[["name"]]$  if  $x$  is recursive and `NULL` otherwise. Partial matching is performed on the names of  $x$  when assignment or replacement is not being done; thus to extract a component of a list, you only need to give enough of the name to make it unique. Replacement of the name component may coerce an object to be a list.

For replacements,

```
x[i] <- value
```

the rule is that the length of  $x$  will be set to the largest value in the indices, if that is bigger than the current length of  $x$ . If  $x$  and `value` do not have the same mode, then one or the other will be coerced to the common mode that can represent both without loss of information. This may mean that replacing elements of an object will change its mode.

## WARNING

To replace a component of a list you must give the entire name of the component. If you abbreviate the name (as you can when merely extracting it), then a new component with the abbreviated name will be added.

## EXAMPLES

```
x[x!=999.999]      # x values not equal to 999.999
```

```
x[order(y)]       # sort x by increasing values of y
```

```
x[-c(1,3)]        # all but the first and third
```

```
list(1:10,2:3)[2] # value is list(2:3)

list(1:10,2:3)[[2]] # value is 2:3

x[2,3] <- 8.4 # change the value of a matrix element

state.x77["Alabama",] # print the row for Alabama

A <- array(1:30, c(5,3,2) ) # array with dimension 5 x 3 x 2

A[] # identical to A

A[1,1,1] # a scalar, the first data value of A

A[1] # the same

A[,1:2,] # a (5,2,2) array

A[A>3] # the vector 4:30
```

## 1- Subscript a Data Frame Object

### DESCRIPTION

Allows the user to extract or replace values from a data frame object by using subscripts.

### USAGE

```
x[j]
x[j] <- value
x[[j]]
x[[j]] <- value
x[i, j, drop=<<see below>>]
x[i, j] <- value
x[[i, j]]
x[[i, j]] <- value
```

### REQUIRED ARGUMENTS

x an object inheriting from class "data.frame".

### OPTIONAL ARGUMENTS

$i, j$  subscript expressions, indicating which elements to extract or replace. The expressions may be empty (meaning all possible subscripts), logical, numeric, or character. If a single argument is given and that argument is not a matrix or list, e.g.

$$x[j]; x[[j]]$$

then  $x$  is treated as a frame or list, and the index  $j$  is assumed to index the variables of the data frame. Some special cases when  $j$  is a list or matrix are described in the "DETAILS" section below. If the subscripting looks matrix-like, e.g.

$$x[i,j]; x[i,]; x[,j]$$

then  $i$  and  $j$  apply to the rows and columns (variables) respectively, and the methods treat  $x$  as a matrix.

**drop** optional flag to control dropping of individual rows or columns. By default, single columns are dropped to the corresponding variable, but single rows remain data frames. If `FALSE` then single columns remain in a dataframe. If `TRUE` then single rows become ordinary lists.

**value** replacement value for the relevant piece of the data frame. This can be a data frame itself, and this is recommended if you are replacing data in more than one variable, unless `value` is a constant. The value can also be an atomic vector or a list. For double subscripts the value should not be a list or data frame.

## VALUE

the data frame formed by extracting or replacing the relevant rows and/or columns. In the case of  $x[i,]$  or  $x[,j]$  selecting a single row or column, the extracted data may be dropped from a data frame to a list or a variable, respectively.  $x[[j]]$  returns a single variable.

## HINTS

To select a single variable from a data frame (for example, in a loop),  $x[[j]]$  is faster than  $x[,j]$ .

Using  $x[i,]$  or  $x[,j]$  can be slow if there are many rows and duplicate values in  $i$ , because creating unique row names is slow. If `attr(x, "dup.row.names")` is not `NULL`, then row names may contain duplicates and subscripting is faster.

## DETAILS

Arguments are passed to these functions by position, not name, with the exception of `drop`, which must be passed by name.

Assigning one or more columns of a data frame will check that the new values have the right number of rows and generally try to ensure that the assignment leaves the data

frame in a valid state. In `x[i,j] <- value`, the characteristics (class, mode, dimension, attributes) of variables in the result depend on the characteristics of the variables in both `x` and `value`. In `x[[j]] <- value`, `x[j] <- value`, and `x[,j] <- value` they depend solely on `value`.

For dataframes containing matrices, these methods handle column subscripts as if each matrix were a single column. For example, if `Y` has 3 elements of which the third is a matrix with 5 columns, `Y[,3]` is (dropped to) a matrix with 5 columns, and `Y[,4]` is undefined. In `x[i,j] <- value`, if `x` contains matrices then (1) `i` must refer solely to existing rows (in other cases, the dimensions of the data frame are expanded as necessary), (2) elements of `value` are assigned to columns of `x` without regard to the dimensions of matrices in `x`. We recommend that either `value` be a data frame with matrices the same dimensions as those in `x[i,j]`, or that double subscripting be used for one variable at a time, e.g. `Y[[3]][i,] <- 2*Y[[3]][i,]`.

A special case for `x[j]` is when `j` is a matrix, either logical with the same dimensions as `x` or numeric with two columns, where the first column refers to rows and the second to column numbers. Then all columns of `x` are coerced to a common data type and matrix variables are converted to multiple columns before subscripting with matrix `j`.

A special case for `x[[j]]` is when `j` is a numeric vector or a list with character or numeric components of length 1. Then `x[[j]]` is equivalent to:

```
x [[ j[[1]] ]] [[ j[[2]] ]] ... [[ j[[length(j)]] ]]
```

For example `x[[1:2]]`, `x[[2]][[1]]`, and `x[[2,1]]` are equivalent.

Attributes of `x` other than `row.names` and `dup.row.names` are lost if columns are subscripted, e.g. `x[j]`, `x[,j]`, `x[i,j]`.

## EXAMPLES

```
solder[sample(900,10), ]
```

```
attach(fuel.frame); fuel.frame[, "h"] <- Weight/Disp.
```

**អថេរចៃដន្យ និង ច្បាប់នៃប្រូបាប៊ីលីតេ  
(Random variables and probability distribution)**

**1- អថេរចៃដន្យ :**

នៅក្នុងការបោះគ្រាប់ឡកឡាក់យើងពុំអាចកំណត់បានជាមុនថា លេខមួយណាកើតមុនទេ ព្រោះវាទាក់ទងទៅនឹងលក្ខខណ្ឌជាច្រើន ។ នៅក្នុងន័យនេះលេខ 1,2,3,4,5,6 ជាតំលៃលេខដែលអាចកើតឡើងដោយចៃដន្យ ។

**និយមន័យ 1:**

អថេរចៃដន្យ (random variable) គឺជាអថេរដែលនៅក្នុងលទ្ធផលនៃវិញ្ញាសានិមួយៗ អាចទទួលបានតំលៃតែមួយគត់ដែលជាតំលៃមួយពុំអាចកំណត់បានជាមុនហើយដែលទាក់ទងទៅនឹងលក្ខខណ្ឌចៃដន្យជាច្រើន ។

**ឧទាហរណ៍ :**

ក្នុងការបោះគ្រាប់ឡកឡាក់បើសិនណាយើងទទួលបានលេខ 1 ក្នុងការបោះលើកទីមួយ យើងនិយាយថា លេខ 1 ជាអថេរចៃដន្យក្នុងការពិសោធន៍លើកទីមួយ ។ ការបោះគ្រាប់ឡកឡាក់បន្តទៀតអាចទទួលបានតំលៃលេខជាហូរហែរ តំលៃទាំងនេះអោយឈ្មោះថា **អថេរចៃដន្យ** ។ ដូច្នេះក្នុងការបោះគ្រាប់ឡកឡាក់អថេរចៃដន្យអាចមាន 6 ករណី : { 1,2,3,4,5,6 } ។

**និយមន័យ 2:**

អថេរចៃដន្យដាច់ (discrete random variable) គឺជាអថេរចៃដន្យដែលអាចទទួលបានតំលៃដាច់ដោយឡែកពីគ្នា ហើយមានប្រូបាប៊ីលីតេមួយកំណត់ ។ ចំនួនតំលៃលេខដែលអាចនៃអថេរចៃដន្យដាច់ជាចំនួនអាចរាប់អស់ ឬ ជាចំនួនរាប់មិនអស់ ។

**ឧទាហរណ៍ :** ចំនួនក្មេងប្រុសដែលទើបនឹងកើតក្នុងចំក្មេងដែលទើបនឹងកើត 200 នាក់ ជាអថេរ ចៃដន្យដែលតាងដោយ X ហើយអថេរចៃដន្យអាចទទួលបានតំលៃ 0,1,2,3,...,200 ។

**និយមន័យ 3 :**

អថេរចៃដន្យជាប់ (continuous random variables) គឺជាអថេរចៃដន្យដែលអាចទទួលបាននូវគ្រាប់តំលៃនៅក្នុងចន្លោះមួយកំណត់ ឬ មួយក្នុងចន្លោះមួយមិនកំណត់ ។ ចំនួនតំលៃនៃអថេរចៃដន្យជាប់ជាចំនួនរាប់មិនអស់ ។

**ឧទាហរណ៍ :**

ចំងាយចរនៃការបាញ់គ្រាប់កាំភ្លើងមួយគ្រាប់គឺជាអថេរចៃដន្យជាប់ ។ ជាការពិតណាស់ចំងាយចរនៃគ្រាប់កាំភ្លើងនេះវាទាក់ទងទៅនឹងលក្ខខណ្ឌជាច្រើន : កំលាំងធាក់នៃកាំភ្លើង ទិសដៅនៃខ្យល់ និងសម្ពាធបរិយាកាស ផ្សេង ៗ ទៀត ដែលយើងពុំអាចគិតទុកជាមុន ។ តំលៃដែលអាចនៃអថេរចៃដន្យនេះស្ថិតនៅក្នុងចន្លោះ (a,b) ។ យើងឃើញថាគ្រាប់កាំភ្លើងអាចធ្លាក់ដោយចៃដន្យនូវគ្រាប់ចំនុចនៃចន្លោះ (a,b) ។

**2- ឧទាហរណ៍ :របាយប្រូបាប៊ីលីតេ (probability distribution) នៃចំនួនអាំងឡែប៊ែរ (switch) ដែលខូច ។**

សហគ្រាស Microtek បានផលិត switch ដែលមានពន្លឺ ។ សំរង់ស្ថិតិមួយបានបង្ហាញមាន 5% នៃ switch ដែលផលិតដោយ Microtek ត្រូវខូចប្រើប្រាស់ពុំបាន ។ សន្មតថាគេជ្រើសរើសដោយចៃដន្យនូវ switch ចំនួន 2 ។ សន្មតថាអថេរចៃដន្យតាងដោយ X : ចំនួន switch ដែលខូចនៅក្នុងគំរូស្ថិតិដែលគេធ្វើការដកស្រង់នេះ ។

ដើម្បីបង្កើតរបាយប្រូបាប៊ីលីតេ នៃអថេរចៃដន្យនេះយើងត្រូវស្គាល់ :

- ក- តំលៃអថេរចៃដន្យទាំងនេះ
- ខ- តំលៃប្រូបាប៊ីលីតេនៃអថេរនីមួយៗ

ជាដំបូងយើងត្រូវកំណត់នូវគ្រាប់ព្រឹត្តិការណ៍ឯកធាតុនៃវិញ្ញាសានេះ ដោយតាង D ជា " switch ដែលខូច " និង B ជា " switch ដែលល្អ " ដូច្នេះព្រឹត្តិការណ៍ទាំងនេះគឺ :

$$\Omega = \{BB, BD, DB, DD\}$$

ព្រឹត្តិការណ៍ឯកធាតុ	តំលៃ X ដែលទាក់ទងទៅនឹងព្រឹត្តិការណ៍ទាំងនេះ
BB	0
BD	1
DB	1
DD	2

យើងដឹងថាមាន switch ចំនួន 5% ខូច (95% អាចប្រើប្រាស់បាន) ដូច្នេះប្រូបាប៊ីលីតេនៃ switch ដែលខូចស្មើ 0.05 និងប្រូបាប៊ីលីតេនៃ switch ដែលអាចប្រើប្រាស់បានស្មើ 0.95 ។ ដោយសន្មតិព្រឹត្តិការណ៍ខាងលើមិនទាក់ទងគ្នា :

$$P(BB)=P(B)*P(B)=0.95*0.95=0.9025$$

$$P(BD)=P(B)*P(D)=0.95*0.05=0.0475$$

$$P(DB)=P(D)*P(B)=0.05*0.95=0.0475$$

$$P(DD)=P(D)*P(D)=0.05*0.05=0.0025$$

ដូច្នេះយើងអាចសង្ខេបតារាងរបាយប្រូបាប៊ីលីតេដូចខាងក្រោម :

ព្រឹត្តិការណ៍ឯកធាតុ	តំលៃ $x_i$ នៃអថេរចៃដន្យ X	ប្រូបាប៊ីលីតេ
$P(X=x_i)$		
BB	$x_1=0$	$P(X=0)=0.9025$
BD	$x_2=1$	$P(X=1)=0.0950$
DB		
DD	$x_3=2$	$P(X=2)=0.0025$
		ផលបូក = 1

ដូច្នេះអថេរចៃដន្យទាំងនេះជាអថេរចៃដន្យដាច់ដែល :

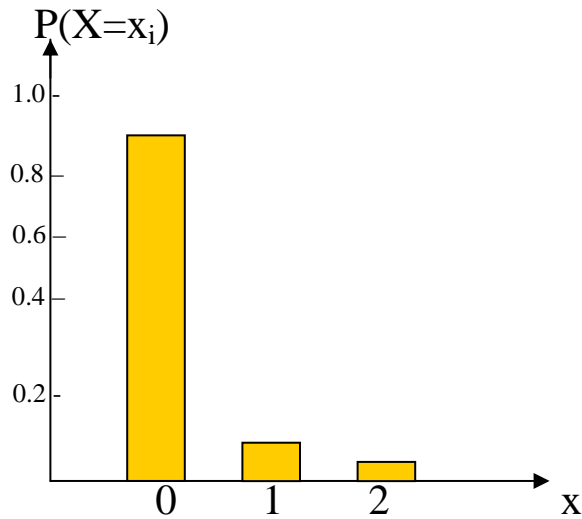
$$X : 0 \quad 1 \quad 2$$

### 3- ក្រាហ្វិកនិងច្បាប់នៃប្រូបាប៊ីលីតេ (Graphics and probability distribution)

#### ដ្យាក្រាមដំបង (bar chart) :

គេអាចតាងរបាយប្រូបាប៊ីលីតេនៃអថេរចៃដន្យដាច់ដោយដ្យាក្រាមដំបង ។ ដោយអនុវត្តលើឧទាហរណ៍ខាងលើយើងអាចបានដ្យាក្រាមដំបងដូចខាងក្រោម ដោយប្រើកម្មវិធី S-plus :

```
function()
{ x<-c(0,1,2)
  probability<-c(0.9025,0.0950,0.0025)
  barplot(probability,names=levels(factor(x)))
  return()
}
```



$$\sum_i P(X = x_i) = 1$$

### អនុគមន៍របាយ (Distribution function)

អនុគមន៍របាយកំណត់ដោយ :  $F(x_i) = P(X \leq x_i)$

នៅក្នុងករណីនៃអថេរចៃដន្យដាច់ :

$$F(x_i) = P(X \leq x_i) = P(X = x_1) + P(X = x_2) + \dots + P(X = x_i)$$

ក្រាហ្វិកនៃអនុគមន៍របាយជាអនុគមន៍កាំជណ្តើរ (step function) ។

### ឧទាហរណ៍ :

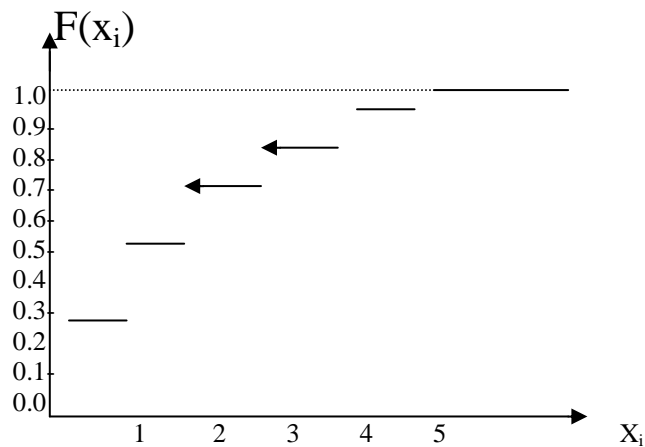
នៅក្នុងក្រុមហ៊ុន sangamex គេបានសិក្សាទៅលើកំហុសនៃការតំឡើងឧបករណ៍ electronic ដែលកំណត់ក្នុងតារាងខាងក្រោម :

ចំនួននៃកំហុស $x_i$	$P(X=x_i)$
0	0.30
1	0.25
2	0.18
3	0.14
4	0.10
5	0.03

ក- កំណត់អនុគមន៍រហាយ និងសង់ក្រាហ្វិក :

ក្នុងន័យនេះយើងត្រូវកំណត់  $P(X \leq x_i) = F(x_i)$  នៃអថេរចៃដន្យ " ចំនួនកំហុសទៅតាមឧបករណ៍ electronic នីមួយៗ " ។

ចំនួនកំហុស $x_i$	$F(x_i) = P(X \leq x_i)$
0	0.30
1	0.55
2	0.73
3	0.87
4	0.97
5	1.00



ខ- កំណត់ប្រូបាប៊ីលីតេដើម្បីអោយចំនួនកំហុសនៃការតំឡើងឧបករណ៍ electronic នេះតូចជាង ឬ ស្មើ 2 ?

$$F(2) = P(X \leq 2) = 0.73$$

គ- កំណត់ប្រូបាប៊ីលីតេដើម្បីអោយចំនួនកំហុសនៃការតំឡើងឧបករណ៍ electronic នេះធំជាង 1?

$$\begin{aligned} P(X > 1) &= 1 - P(X \leq 1) \\ &= 1 - F(1) \\ &= 1 - 0.55 \\ &= 0.45 \end{aligned}$$

Graph នៃអនុគមន៍រាយខាងលើអាចទទួលបានដោយប្រើកម្មវិធី S-plus :

```
function()
{ x<-c(0,1,2,3,4,5)
  dist.fun<-c(0.30,0.55,0.73,0.87,0.97,1)
  plot(stepfun(x,dist.fun),type="l")
  return()
}
```

**លំហាត់ :**

១- នៅក្នុងការអង្កេតមួយដែលធ្វើឡើងនៅក្នុងហាងលក់ទំនិញដ៏ធំមួយបានអោយដឹងថា ក្នុងចំណោមអតិថិជន 200 នាក់ មាន 120 នាក់បានបញ្ជាក់ថាគាត់មកហាងលក់ទំនិញនេះដោយសារគាត់ឃើញការផ្សាយពាណិជ្ជកម្ម ។ មួយចំនួនទៀតមកទីនេះដោយពុំបានឃើញការផ្សាយពាណិជ្ជកម្មទេ ។ ក្នុងចំណោមអតិថិជនទាំង 200 នាក់នេះដែលត្រូវបានចោទសួរមាន 60 នាក់បានទិញសម្ភារៈមួយចំនួនហើយក្នុងចំណោមអ្នកទាំង 60 នាក់នេះ មាន 20 នាក់បានឃើញការផ្សាយពាណិជ្ជកម្ម ។

ក- ចូរបង្កើតតារាង two-way table

ខ- កំណត់ប្រូបាប៊ីលីតេដើម្បីអោយអតិថិជនដែលពុំបានឃើញការផ្សាយពាណិជ្ជកម្ម បានទិញសម្ភារៈមួយចំនួនក្នុងហាងនេះ ?

គ- កំណត់ប្រូបាប៊ីលីតេដើម្បីអោយអតិថិជនដែលបានឃើញការផ្សាយពាណិជ្ជកម្ម បានទិញសម្ភារៈមួយចំនួនក្នុងហាងនេះ ?



BUILD BRIGH UNIVERSITY, FACULTY OF SCIENCE AND TECHNOLOGY.

Prepared by MEAK KAMERANE( Statistician, Data analyst, Mathematician).