APPENDIX A

Operators in R

OBJECTIVE

This appendix provides a description of selected R built-in functions, as well as a discussion about the use of relational operators. The built-in functions covered, include statistical functions in Table A.1 , text functions in Table A.2 , and mathematical and trigonometry functions in Table A.3 . I added a short list of useful general reference functions in Table A.4 . These are functions that I have often used for manipulating vectors in R. The relational operators are covered in section A.2. You will learn the rules that regulate the comparison between 2 vectors.

Contents

A.1 *Selected Built-In Functions in R* 428
A.2 *Relational Operators* 434

A.1 Selected Built-In Functions in R

Table A.1 : Statistical Functions

max(x)	This function computes the largest of all elements pertaining to object x.
min(x)	This function computes the smallest of all elements pertaining to object x.
mean(`x, trim=0, na.rm=FALSE`)	This function computes the mean of object x. In Excel you will use various versions of the AVERAGE() function.
median(x)	This function computes the median of object x, assumed to contain several numbers.
quantile(x,*probs*)	This function computes one quantile or several quantiles of the numeric vector x, based on the vector of probabilities *probs* (Figure A.1 shows how this function is used)
sd(x)	This function computes the standard deviation of object x elements.
sum(x)	This function computes the sum of all elements pertaining to object x.
var(x)	This function computes the variance of object x elements.
dnorm(`x,mean=0,sd=1`) or **dnorm**(`x,m=0,s=1`)	*Evaluates the Normal density function at all elements of vector x. By default meam=0 and sd=1.*

pnorm(q)	Evaluates the cumulative distribution function (CDF) of the Normal distribution at the probability value(s) q, where q is vector of probabilities (e.g. `pnorm(c(1.96,2.3))` would yield `0.975 0.989`).
qnorm(p)	*Computes the* $100p^{th}$ *quantile of the Normal distribution, where p is a vector of probabilities.*
rnorm(n, m=0,sd=1)	Creates a random sample of n normal variate with mean m and standard deviation sd.
dunif(x, min=0, max=1)	*Evaluates the uniform density function between min and max at all points of x.*
punif(x, min=0, max=1)	Evaluates the uniform cumulative probability distribution between min and max at all points of vector x.
qunif(p, min=0, max=1)	*Evaluates quantiles of uniform distribution between min and max at all values of the probability vector p.*
runif(n, min=0, max=1)	Generates n random uniform variates between min and max.
sample(x, size, replace=FALSE)	*Takes a random sample of specified size from the elements of x with replacement if* ***replace=TRUE*** *or without replacement if* ***replace=FALSE*** *(default value).*

Table A.2 : List of Text Functions

substr(x,start=n1,stop=n2)	Useful function for extracting or replacing substrings in a character vector x. Let x<-"KILEM GWET". Then substr(x,7,10) yields "GWET". The equivalent Excel function would be MID.
strsplit(x, split)	Split the elements of character vector x at split. This function is useful if you want to create a vector of characters out of a string.
paste(v1,v2, sep=" ")	Concatenate 2 vectors v1 and v2 elementwise, after converting them to characters. Note that sep defines the character that separates 2 concatenated elements. This is the R equivalent of the Excel function CONCATENATE().
toupper(x)	Function would convert text in x to uppercase, equivalent to the Excel function UPPER.
tolower(x)	Function would convert text in x to lowercase, equivalent to the Excel function LOWER.

Table A.3 : List of Math and Trigonometry Functions

R Function	Excel Function	Description
abs(x)	**ABS**(x)	Returns the absolute value of object x. In R, x may be a vector or a matrix, and could be a cell or range of cells in Excel.
ceiling(x)	**CEILING**(x,1)	Returns the smallest integer that equals or exceeds x.
cos(x), **cosh**(x)	**COS**(x), **COSH**(x)	Returns the cosine and the hyperbolic cosine of x.
exp(x)	**EXP**(x)	Returns e raised to the power of number. The constant e equals 2.71828182845904, the base of the natural logarithm.
floor(x)	**FLOOR**(x,1)	Returns the largest integer that is equal or smaller than x.
log(x)	**LN**(x)	Returns the natural logarithm of a x, which is based on the constant e = 2.71828182845904.
log10(x)	**LOG10**(x)	Returns the base-10 logarithm of a x.
round(x, digits=n)	**ROUND**(x,n)	Rounds x to n digits after the decimal place. Parameter n is optional in R and mandatory in Excel.
sin(x), **sinh**(x)	**SIN**(x), **SINH**(x)	Returns the sine and the hyperbolic sine of x.
sqrt(x)	**SQRT**(x)	Returns the square root of object x. In R, x may be a vector or a matrix, and could be a cell or range of cells in Excel.
tan(x), **tanh**(x)	**TAN**(x), **TANH**(x)	Returns the tangent and the hyperbolic tangent of x.
trunc(x)	**TRUNC**(x,1)	Truncates the value of x towards 0.

Table A.4 : Reference Functions

R Function	Description
ncol(x)	Number of columns of a data frame or a matrix
nrow(x)	Number of rows of a data frame or a matrix
seq`(from n, to m, by k)`	Generates a sequence of numbers from n to m with increment of k.
rep`(x,times=t,each=e)`	This function replicates the values in x, t times and each element of x is repeated e times.
sort`(x,decreasing=FALSE)`. The `decreasing` parameter is optional.	Sorts all elements of vector x in ascending order by default. If you want to sort x is descending order, then you must specify `decreasing=TRUE`.

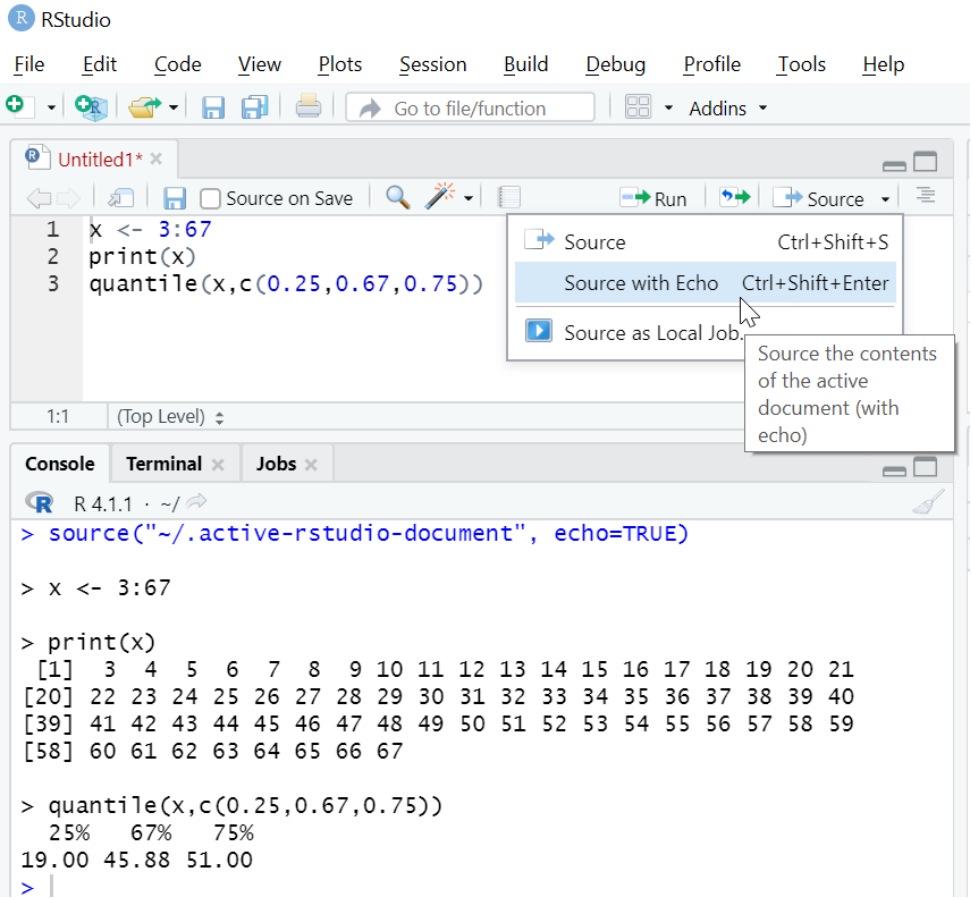

Figure A.1: Computing Quantiles of a Numeric Vector

A.2 Relational Operators

Table A.5 shows the relational operators supported by R language. You can compare 2 vectors `a` and `b` (e.g. `(a>b)`). This operation is possible only if the number of elements of the longer vector is a multiple of the number of elements of the shorter vector. In this case, each element of the shorter vector will be compared with the *Corresponding* element in the longer vector.

Consider the commands `a<-c(5,2)` and `b<-c(1,4,3,6)`. The longer vector `b` has 4 elements whereas the shorter one `a` has only 2. Nevertheless, the binary operator `>` used in the command `(a>b)` can still be applied on vectors of different lengths. This operation is done by *recycling* the elements of the shorter vector `a`. The recycling process consists of extending the shorter vector until it matches the length of the longer one. The elements of the short vector are repeated sequentially, starting by the first one. R will evaluate `(a<b)` according to the following table:

a[a]	5	2	5	2
b	1	4	3	6
(a>b)	TRUE	FALSE	TRUE	FALSE

[a] Extended version of vector `a`.

It follows from this table that 5 (the first element of `a`) is associated the 1 and 3 (the first and third element of `b`), whereas 2 (the second element of `a`) is associated with 4 and 6, the second and fourth element of `b`). The command `(a>b)` produced the boolean vector `(TRUE, FALSE, TRUE, FALSE)`.

Note that if the number of `b`-elements is not a multiple of the number of `a`-elements, some `a`-elements will be recycled more than others, which will generate a warning message. But recycling will still take place. Recycling is commonly-used in R for binary element-wise operators on vectors.

Table A.5 : Relational operators for comparing 2 vectors a and b

Operator	Description	Example
$>$	**Greater Than:** The command `a>b` checks if each element of `a` is greater than the *corresponding* element of `b` or not.	`print(c(3.5,7.1)>4.3)` will produce the following vector: `[1] FALSE TRUE`
$<$	**Smaller Than:** The command `a<b` checks if each element of `a` is smaller than the corresponding element of `b` or not.	`print(c(3.5,7.1)<4.3)` will produce the following vector: `[1] TRUE FALSE`

$==$	**Equal To:** The command `a==b` checks if each element of `a` is equal to the corresponding element of `b` or not.	`a <- c(3.5,7.1,4.3)`
		`b <- c(1.5,7.0,4.3)`
		`print(a==b)`
		produces vector:
		`[1] FALSE FALSE TRUE`

$<=$	**Smaller or Equal To:** The command `a<=b` checks if each element of `a` is smaller than or equal to the corresponding element of `b` or not.	`a <- c(3.5,7.1,4.3)`
		`b <- c(1.5,7.0,4.3)`
		`print(a<=b)`
		produces vector:
		`[1] FALSE FALSE TRUE`

$>=$	**Greater or Equal To:** The command `a>=b` checks if each element of `a` is greater than or equal to the corresponding element of `b` or not.	`a <- c(3.5,7.1,4.3)`
		`b <- c(1.5,7.0,4.3)`
		`print(a>=b)`
		produces vector:
		`[1] TRUE TRUE TRUE`

$!=$	**Not Equal To:** The command `a!=b` checks if each element of `a` is different from the *corresponding* element of `b` or not.	`a <- c(3.5,7.1,4.3)`
		`b <- c(1.5,7.0,4.3)`
		`print(a!=b)`
		produces vector:
		`[1] TRUE TRUE FALSE`