Mathematical Functions

```
x <- c(1, 2, 2, 3, 3, 4, 5, 6, 7, 7, 8, 9, 10, 10)
sum(x)
```

```
## [1] 77
```

```
prod(x)
```

```
## [1] 1524096000
```

```
mean(x)
```

```
## [1] 5.5
```

```
unique(x)  ## eliminates duplicates
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
summary(x)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    3.00    5.50    5.50    7.75   10.00
```

```
median(x)
```

```
## [1] 5.5
```

```
floor(x)
```

```
##  [1]  1  2  2  3  3  4  5  6  7  7  8  9 10 10
```

```
sd(x)  ## standard deviation
```

```
## [1] 3.082207
```

```
sqrt(x) ## square root
```

```
##  [1] 1.000000 1.414214 1.414214 1.732051 1.732051 2.000000 2.236068 2.449490
##  [9] 2.645751 2.645751 2.828427 3.000000 3.162278 3.162278
```

```r
abs(x) ## absolute value
```

```
##  [1]  1  2  2  3  3  4  5  6  7  7  8  9 10 10
```

```r
Inf  ## infinity, also possible in the negative
```

```
## [1] Inf
```

```r
NaN  ## Not a number, such as 0/0
```

```
## [1] NaN
```

```r
sample(1:6, 4, replace = TRUE) ## simulate rolling 4 dice
```

```
## [1] 1 5 5 4
```

```r
sample(letters, 3)  ## letters is a predefined vector in R
```

```
## [1] "d" "u" "l"
```

```r
sample(LETTERS, 4) ## upper case letters
```

```
## [1] "V" "G" "P" "X"
```

```r
log(x)  ## natural log
```

```
##  [1] 0.0000000 0.6931472 0.6931472 1.0986123 1.0986123 1.3862944 1.6094379
##  [8] 1.7917595 1.9459101 1.9459101 2.0794415 2.1972246 2.3025851 2.3025851
```

```r
log10(x) ## log base 10
```

```
##  [1] 0.0000000 0.3010300 0.3010300 0.4771213 0.4771213 0.6020600 0.6989700
##  [8] 0.7781513 0.8450980 0.8450980 0.9030900 0.9542425 1.0000000 1.0000000
```

```r
log(x, base = 2) ## log base 2
```

```
##  [1] 0.000000 1.000000 1.000000 1.584963 1.584963 2.000000 2.321928 2.584963
##  [9] 2.807355 2.807355 3.000000 3.169925 3.321928 3.321928
```

```r
exp(x) ## e to the power of x
```

```
##  [1]     2.718282     7.389056     7.389056    20.085537    20.085537
##  [6]    54.598150   148.413159   403.428793  1096.633158  1096.633158
## [11]  2980.957987  8103.083928 22026.465795 22026.465795
```

```r
table(x)  ## creates a table that counts repeated elements
```

```
## x
##  1  2  3  4  5  6  7  8  9 10
##  1  2  2  1  1  1  2  1  1  2
```

```r
seq_len(8)  ## creates a sequence
```

```
## [1] 1 2 3 4 5 6 7 8
```

```r
seq_along(x)  ## numeric vector with sequence equal to length(x)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14
```

```r
      ## identical to seq_len(length(x))

quantile(x, c(0, 0.25, 0.5, 0.75, 1))  ## quartiles, percentiles, deciles...
```

```
##    0%   25%   50%   75%  100%
##  1.00  3.00  5.50  7.75 10.00
```

```r
signif(x) ## rounds to significant figures
```

```
##  [1]  1  2  2  3  3  4  5  6  7  7  8  9 10 10
```

```r
sample(x, size = 1) ## takes a sample of SIZE from elements of x
```

```
## [1] 3
```

```r
rep(2, 4)  ## repeats a number, n amount of times
```

```
## [1] 2 2 2 2
```

```r
gl(3, 10)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
## Levels: 1 2 3
```

```r
 ## generates a numeric vector, elements are 1 repeated k times,
 ## then 2 repeated k times, then 3, until the number is equal
 ## to n.
rnorm(10, 5, 2.1)
```

```
##  [1] 8.777943 5.756720 4.492437 3.699554 4.977941 5.072410 6.030203 3.913923
##  [9] 4.827571 7.566738
```

```
  ## generates n random numbers with mean and standard dev, based on a
  ## normal distribution.
dnorm(-0.5, 0, 1)
```

```
## [1] 0.3520653
```

```
  ## calculates the density. I.e.: for a value of `x`, the function gives
  ## the frequency (vertical axis) of the distribution. Equivalent of
  ## applying the normal distribution function:
```

$$\frac{1}{\sigma\sqrt{2\pi}} * e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

```
pnorm(-0.5, 0, 1)
```

```
## [1] 0.3085375
```

```
  ## returns the area under the curve, from $-\infty$ to `q`. `q` is a
  ## z-score. (I.e. the probability that any given value of the population
  ## falls between $-\infty$ and `q`.)
qnorm(-1.2, 0, 1)
```

```
## Warning in qnorm(-1.2, 0, 1): NaNs produced
```

```
## [1] NaN
```

```
  ## is the opposite of `pnorm`. Given a probability `p`, it returns the
  ## z-score for that probability.

runif(10, 3, 13)
```

```
##  [1]  9.419072 10.586491  6.975940  8.466788  7.097755 10.201285  9.379966
##  [8]  4.122051  8.776438  4.739824
```

```
  ## generates n random numbers, with minimum and max set, based on a
  ## uniform distribution. d, p and q functions also exist for this
  ## distribution
rpois(10, 30)
```

```
##  [1] 27 28 23 31 28 26 24 37 28 33
```

```
  ## generates n random numbers, with rate lambda, based on a poisson
  ## distribution. d, p and q functions also exist for this
  ## distribution.
rbinom(5, size = 100, prob = 0.25)
```

```
## [1] 20 25 26 22 15
```

```
  ## generates 100 coin tosses with 0.25 probability of success and
  ## sums up the successes, the repeats the proces 5 times and
  ## outputs a vector or the results. d, p and q functions also exist
rexp(5)  ## exponential
```

```
## [1] 1.6997006 0.4662303 0.4192887 0.6716409 1.6055194
```

```
rchisq(5, 1)  ## chi-squared
```

```
## [1] 0.22822516 0.11916310 0.08058733 0.51470668 2.75510368
```

```
##rgamma()


cor(x, runif(length(x), 0, 10))
```

```
## [1] -0.02081856
```

```
  ## calculates correlation between to vectors

m <- matrix(c(5, 1, 0,
              3,-1, 2,
              4, 0,-1), nrow=3, byrow=TRUE)
solve(m) ## calculates the inverse matrix
```

```
##        [,1]    [,2]   [,3]
## [1,] 0.0625  0.0625  0.125
## [2,] 0.6875 -0.3125 -0.625
## [3,] 0.2500  0.2500 -0.500
```

```
m %*% solve(m)  ## mathematical matrix multiplication
```

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

```
## svd()  ## single value decomposition of a matrix
```

```
rep <- replicate(8, rbinom(5, 25, prob = 0.2))
rep  ## can create a matrix for results
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
## [1,]    3    4    7    5    6    6    1    9
## [2,]    3    3    3    4    6    2    5    3
## [3,]    7    3    3    6    6    5    3    4
## [4,]    6    6    8    6    4    6    5    4
## [5,]    6    5    8    6    2    5    5    6
```

```r
colMeans(rep) ## means of each column
```

```
## [1] 5.0 4.2 5.8 5.4 4.8 4.8 3.8 5.2
```

```r
hist(colMeans(rep)) ## generate a histogram
```