# R Programming Basics - Useful Builtin Functions for Statistics

## Vectorized Arithmetic - most arithmetic operations in R work on vectors.

Here are a few commonly used summary statistics.

Please try to use them, e.g. to each column of a dataframe.

```r
testVect = c(1,3,5,2,9,10,7,8,6)

min(testVect) # minimum
```

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

```r
max(testVect) # maximum
```

```
## [1] 10
```

```r
mean(testVect) # mean
```

```
## [1] 5.666667
```

```r
median(testVect) # median
```

```
## [1] 6
```

```r
var(testVect) #variance
```

```
## [1] 10
```

```r
sd(testVect) # standard deviation
```

```
## [1] 3.162278
```

```r
# HEY YOU CAN GENERATE VECTORS BY selecting the column of datasets !
# if cars is a dataframe with a column named speed :
vect1 = cars$speed
# another example
vect2 = cars$dist

# now you can treat each vector as an empirical distribution
# and apply your statistical functions:
quantile(vect1) # sample quantiles
```

```
##   0%  25%  50%  75% 100%
##    4   12   15   19   25
```

```r
summary(vect1)  # same thing, plus the mean
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     4.0    12.0    15.0    15.4    19.0    25.0
```

```r
cov(vect1,vect2) # sample covariance between vect1 and vect2
```

```
## [1] 109.9469
```

```
cor(vect1,vect2) # sample correlation coefficient
```

```
## [1] 0.8068949
```

## Calculations on probability distributions - area under curve, percentiles, random sampling

### distributions

```
- *binom* : Binomial Distribution (Pascal)
- *pois*  : Poisson Distribution (Poisson)
- *unif* :  Uniform Distribution
- *exp* : Exponential Distribution
- *norm* : Normal Distribution (Gauss, Laplace, Legendre)
- *chisq* : Chi-Squared Distribution (Pearson)
- *t*   : t Distribution (Student)
- *f*  : F Distribution (Fisher)
```

### Why do we mean by distributions?

Load the course summary (main index) in your browser and go to the 'Supplementary material section'.

Load and read Hey, let's talk about distributions!

This material is for fun. Nothing there is required for the course.

### key prefix letters

```
# *d* : probability density function/probability mass function.
# *p* : cumulative distribution function.
# *q* : quantile function. Returns percentiles, also known as quantiles.
# *r* : random number generation
```

### Explaining pdf, cdf and so on

q (quantile): What is the value of the OUTCOME (observed quantity) that 'corresponds' to a GIVEN probability (input for quantile)

for example: what is the 80% quantile of canadian men heights?

If $X$ is the random value of the observation (i.e. the measured height of a randomly sampled canadian man), we are looking for (a R function that gives) the value of the height $H$ such that $P(X < H) = 0.80$. What is your guess for $H$? How would you estimate it?

EXAMPLE

Assume that $X \sim N(1.7, 0.5)$

How do I use R to calculate the value $q$ such that $P(X < q) = 0.8$?

```
sdval=sqrt(0.5)
qnorm(0.8,mean=1.7,sd=sdval)
```

```
## [1] 2.295116
```

Is this a credible result for human height? Why?

p (probability): This is the reverse of quantiles. We give the outcome (i.e. the height) and compute the PROBABILITY THAT THE OUTCOME is less then H. For example, we give the height: $H = 1.781m$ and we want an algorithm (R function) that calculates $P(X < 1.781cm)$. In 2014, this was about 0.5. The probability that a random variable is less than the given numerical value $x$ is called the cumulative distribution function: $F(x) = P(X < x)$.

EXAMPLE:

Say $X \sim Norm(0, 1)$

How do I use R to calculate $P(X < 1.5)$?

```
pnorm(1.5)
```

```
## [1] 0.9331928
```

Geometric interpretation (AUC):

d (probability density function): This is is the derivative of the cumulative density function. This function $f(x)$ can be thought of as the probability of realizing an outcome in the immediate vicinity of the numerical value $x$, i.e. $f(x)dx = P(X \in [x, x + dx])$.

EXAMPLE:

Suppose that $X \sim Binom(N = 4, p = 0.3)$

What is the probability that $X = 3$?

```
dbinom(3,size=4,prob=0.3)
```

```
## [1] 0.0756
```

How would you check that this is correct?‘

r (random sample): Here we are looking for a way to PRODUCE A NEW OBSERVATION, i.e. a new height for another (randomly sampled) man. We must choose some model (a generative distribution model) that represents the relative importance of men heights in the canadian male population, and must find an algorithm to sample (i.e. produce a new example of output) from this distribution. R does this for us, for a couple of well-known distributions.

EXAMPLE:

In my house, I have a smoke detector.

This detector uses the radioactive decay of a small amount of Americium (Nova-Scotium to be more exact) to detect the presence of smoke (which disrupts the electrical contact caused by radioactive decay in a small ion chamber)

The decay reaction (just for the curious) is:

241Am $\rightarrow$ 237Np + 4He

Assuming that alpha particles are produced at an average rate of 0.8392 emissions by second, and that the count of emissions in intervals of 5 seconds obey a Poisson distribution, how would I ask R to produce a sequence of random counts that mimics the production of alpha particles in my smoke detector for 15 intervals?

Can you try to use the rpoisson function to do this?

```
#rpois(...)
```

This exercise is called : random sampling from a given distribution (here: a Poisson distribution).

## HEY BUT CAN R FUNCTIONS REALLY TELL ME THE PROBABILITY OF EVENTS in my dataset?

No.

The R functions make ideal calculations based on mathematical models.

This may differ from the 'truth' because in many cases, we do not know from what distributions the data came from.

However, probability theory and statistics provide ways to estimate these distributions and the targets are often the model distributions mentioned here.

Let us take an example.

Suppose that $X$ is the height of a canadian male in 2014. Will R tell me that $P(X < 1.781cm) = 0.5$? Absolutely not. R does not know the distribution. You will have to tell R what kind of distribution to use. If you pretend (very good approximation, see live histograms or the work of Sir Francis Galton on human stature) that $X$ comes from or follows a normal distribution, then you will have to provide the value of the parameters: mean and variance of the distribution. Finally, you will just ask R to calculate *dnorm* for those parameters.

## HOW CLOSE are real distributions from model distributions?

For an answer, see the supplementary material (Let's talk about distributions!)

## Examples of manipulation of the statistical distributions in R

- What is the area under the standard normal curve to the left of 1.5?

```
pnorm(1.5)
```

```
## [1] 0.9331928
```

- What point cuts off an area .975 to its left under the standard normal curve? this point also cuts off an area .025 to its right under the standard normal curve.

```
qnorm(.975)
```

```
## [1] 1.959964
```

- What is the area under the standard normal curve to the left of 1.96? To the left of -1.96? What is the area under the curve betwen -1.96 and 1.96?

```
pnorm(1.96)
```

```
## [1] 0.9750021
```

```
pnorm(-1.96)
```

```
## [1] 0.0249979
```

```
pnorm(1.96)-pnorm(-1.96)
```

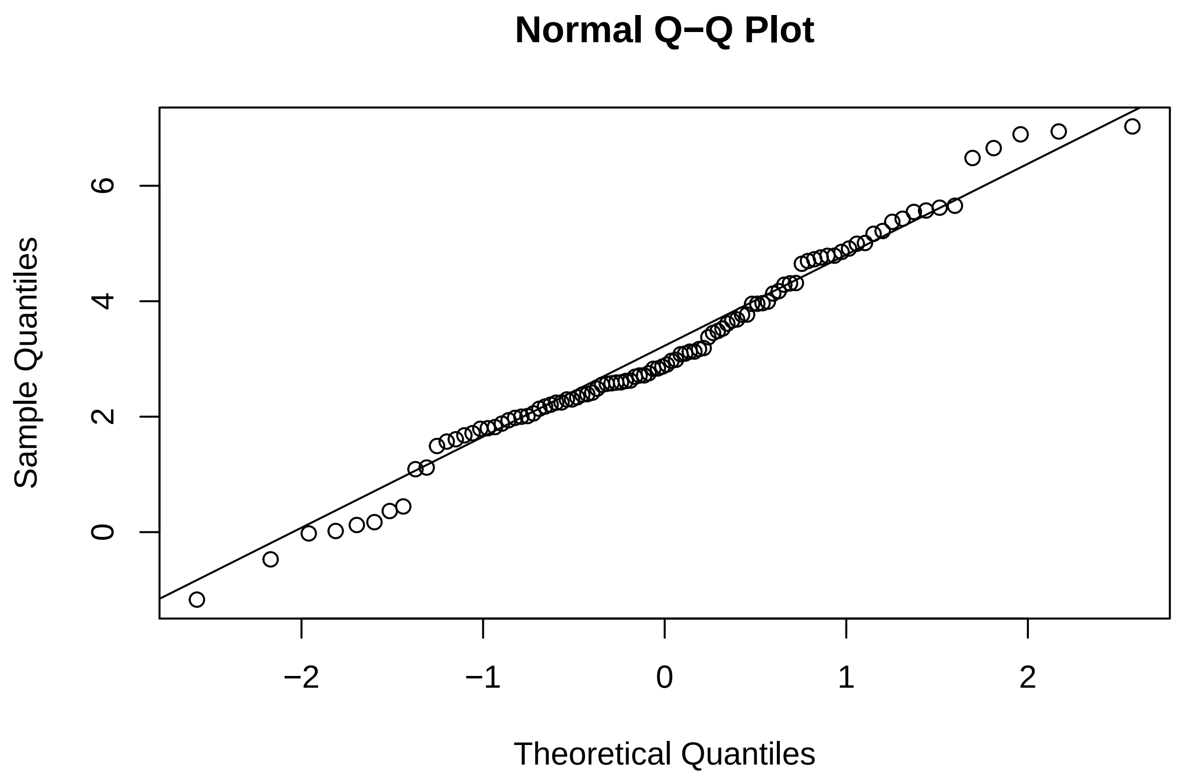
```
## [1] 0.9500042
```

- Simulate 100 observations from the normal distribution with mean $\mu = 3$ and standard deviation $\sigma = 1.5$. Calculate some summary statistics for the data, and also show a normal QQ plot, which can be used to assess normality.

```
data=rnorm(100,mean=3,sd=1.5)
summary(data)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -1.169   2.166   2.885   3.150   4.292   7.028
```

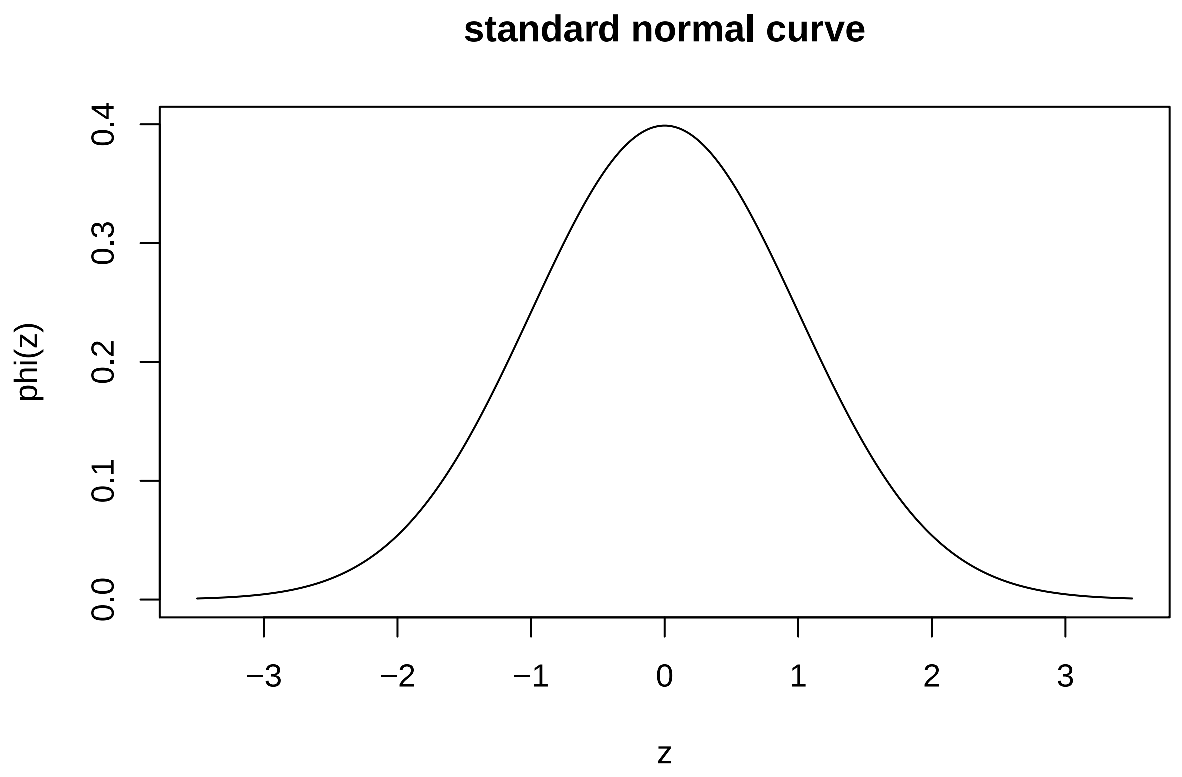
```
qqnorm(data)
qqline(data)
```



- Plot the standard normal density function $\phi(z)$ from $z = -3.1$ to $z = 3.1$, where

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-.5z^2}$$

```
z=seq(from=-3.5, to=3.5, length.out=300)
dz=dnorm(z)
plot(z,dz,xlab="z",ylab="phi(z)",main="standard normal curve",type="l")
```



- What is the probability that a normal random variable with mean 4 and standard deviation 1.5 is greater than 2.5?

```
1-pnorm(2.5,4,1.5)
```

```
## [1] 0.8413447
```

- What is the probability that a $t$ random variable with 12 degrees of freedom is less than 2? Greater than 2?

```
pt(2,12)
```

```
## [1] 0.9656725
```

```
1-pt(2,12)
```

```
## [1] 0.03432751
```

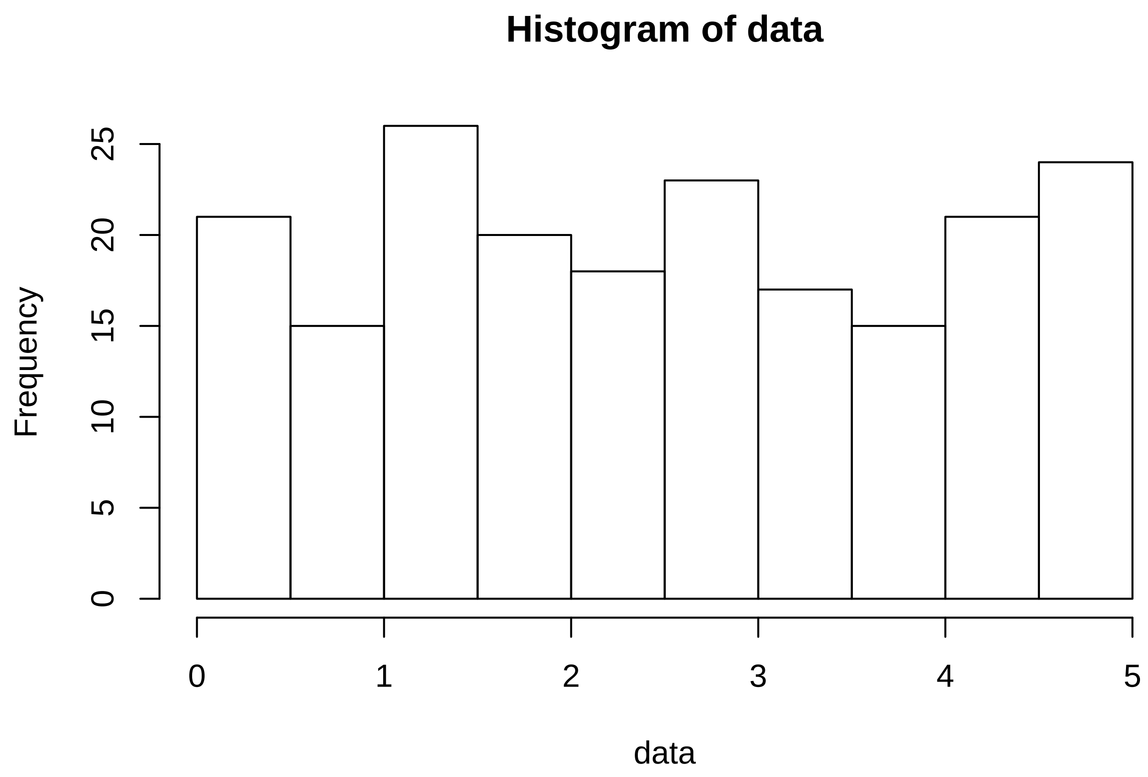
- What are the upper and lower 2.5'th percentile of a $t$ random variable with 6 degrees of freedom?

```
qt(c(.025,.975),6)
```

```
## [1] -2.446912  2.446912
```

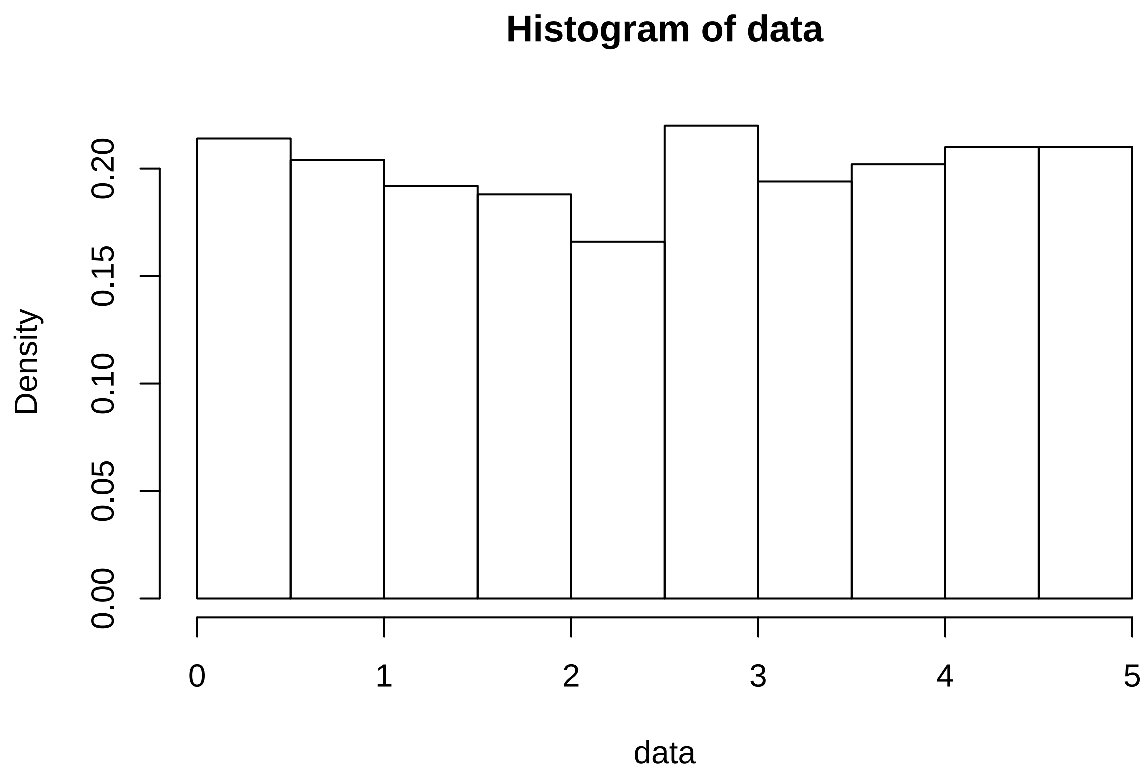
- Simulate 200 observations from the uniform distribution on the interval (0,5). Make a histogram of the data with 10 bars.

```
data=runif(200,0,5)
hist(data,nclass=10)
```



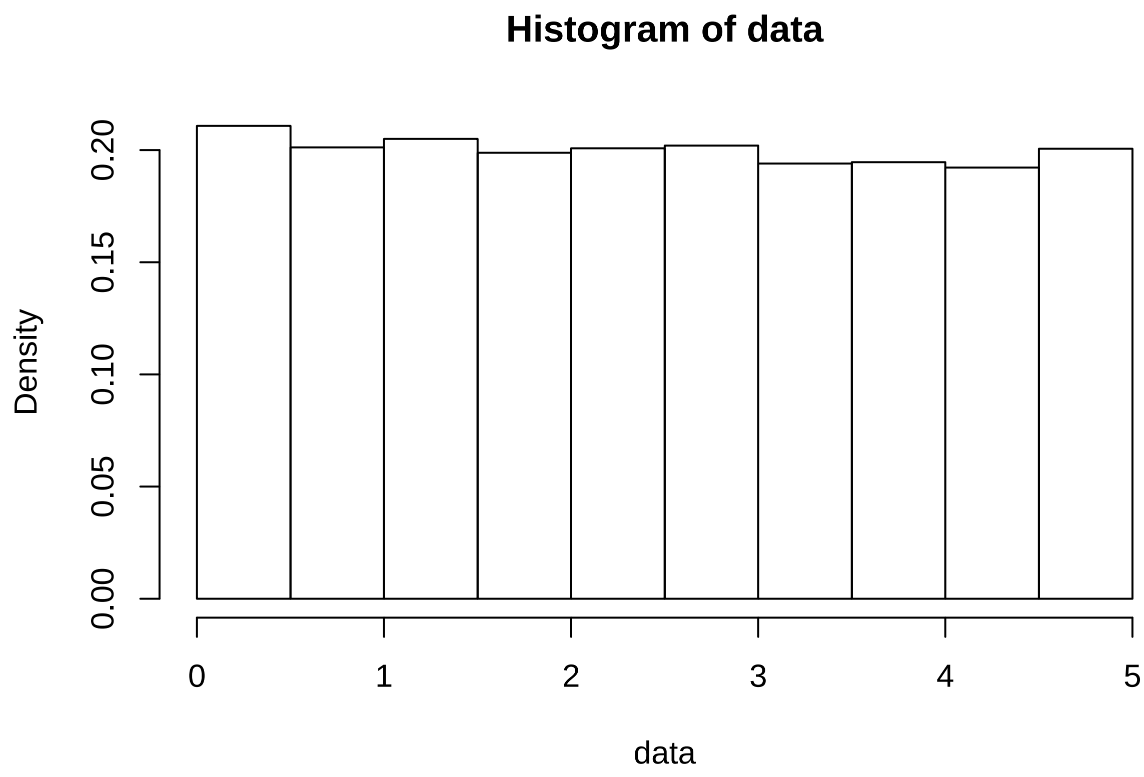
- What about with 1000 observations? Does it look more uniform? Make a probability histogram, which means the bars are normalized so that the total area is 1.

```
data=runif(1000,0,5)
hist(data,nclass=10,prob=T)
```



- What about with 10000? The law of large numbers indicates that the histogram will look close to uniform for a sufficiently large sample size.

```
data=runif(10000,0,5)
hist(data,nclass=10,prob=T)
```



- What is the probability that a binomial random variable with parameters $n$ and $p$ is equal to $x$?

$$p(x) = \binom{n}{x} p^x (1-p)^{n-x}$$

eg. with $n = 10$, $p = .2$, $x = 2$

```
dbinom(2, size=10, prob=.2)
```

```
## [1] 0.3019899
```

- What is the probability that a binomial random variable $X$ with parameters $n$ and $p$ is less than or equal to $x$?

$$P(X \leq x) = \sum_{k=0}^{x} \binom{n}{x} p^x (1-p)^{n-x}$$

eg. with $n = 10$, $p = .2$, $x = 2$

```
pbinom(2, size=10, prob=.2)
```

```
## [1] 0.6777995
```

- Simulate 20 observations from the binomial distribution with $n = 10$ and $p = .2$.

```
data=rbinom(2, size=10, prob=.2)
data
```

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

### When generating random numbers, use *set.seed()* to reproduce results.

```
set.seed(100)
rnorm(5)
```

```
## [1] -0.50219235  0.13153117 -0.07891709  0.88678481  0.11697127
```

```
rnorm(5)
```

```
## [1]  0.3186301 -0.5817907  0.7145327 -0.8252594 -0.3598621
```

```
set.seed(100) # reproduce the results
rnorm(5)
```

```
## [1] -0.50219235  0.13153117 -0.07891709  0.88678481  0.11697127
```

```
rnorm(5)
```

```
## [1]  0.3186301 -0.5817907  0.7145327 -0.8252594 -0.3598621
```

## Useful functions for working on matrices.

### apply - apply a function over rows or columns of a matrix

- syntax: apply(matrixname,dimension,FUN)
- example: for the airquality data set, count the number of missing values by row (for each record), and by column (for each variable)

```
nmiss=function(x){sum(is.na(x))}
apply(airquality,1,nmiss)
```

```
##   [1] 0 0 0 0 2 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 2 0 0 0 0 1 1 1 1 1 1 1
##  [38] 0 1 0 0 1 1 0 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 0 0
##  [75] 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0
## [112] 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [149] 0 1 0 0 0
```

```
apply(airquality,2,nmiss)
```

```
##   Ozone Solar.R    Wind    Temp   Month     Day
##      37       7       0       0       0       0
```

```
mydata=read.csv("http://www-bcf.usc.edu/~gareth/ISL/Auto.csv")

## %*% -inner product.  dot product. matrix multiplication.
```

```
M1=matrix(1:9,byrow=T,ncol=3)
M2=matrix(1:15,byrow=T,ncol=5)
M1
```

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

```
M2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    2    3    4    5
## [2,]    6    7    8    9   10
## [3,]   11   12   13   14   15
```

```
M1%*%M2
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   46   52   58   64   70
## [2,]  100  115  130  145  160
## [3,]  154  178  202  226  250
```

## outer - calculate an outer product between two arrays

```
outer(X, Y, FUN = "*", ...)
```

```
X, Y: First and second arguments for function 'FUN'.  Typically a
      vector or array.
```

FUN: a function to use on the outer products.

The function FUN will be called using each element of X and each element of Y.

Example: The following calculates a table of values x^y, for each x in 1:10, and each value of y in -2:2. Row and column names are added to the table.

```
x=1:10
y=-2:2
powertable=outer(x,y,FUN="^")
rownames(powertable)=x
colnames(powertable)=y
```