In [1]:
```
covar=eye(5);
```

In [2]:
```
for i = 1:5
    for j=1:5
        if i==j
            covar[i,j]=1.0
        else
            covar[i,j]=0.5
        end
    end
end
```

In [3]:
```
covar
```

Out[3]:
```
5x5 Array{Float64,2}:
 1.0  0.5  0.5  0.5  0.5
 0.5  1.0  0.5  0.5  0.5
 0.5  0.5  1.0  0.5  0.5
 0.5  0.5  0.5  1.0  0.5
 0.5  0.5  0.5  0.5  1.0
```

In [4]:
```
meanVec=rand(5)
```

Out[4]:
```
5-element Array{Float64,1}:
 0.92027
 0.379231
 0.474405
 0.866161
 0.305309
```

In [5]:
```
lw=chol(covar,:L)
```

Out[5]:
```
5x5 Triangular{Float64,Array{Float64,2},:L,false}:
 1.0  0.0       0.0       0.0       0.0
 0.5  0.866025  0.0       0.0       0.0
 0.5  0.288675  0.816497  0.0       0.0
 0.5  0.288675  0.204124  0.790569  0.0
 0.5  0.288675  0.204124  0.158114  0.774597
```

In [6]:
```
lw*transpose(lw)
```

Out[6]:
```
5x5 Array{Float64,2}:
 1.0  0.5  0.5  0.5  0.5
 0.5  1.0  0.5  0.5  0.5
 0.5  0.5  1.0  0.5  0.5
 0.5  0.5  0.5  1.0  0.5
 0.5  0.5  0.5  0.5  1.0
```

In [7]:
```
?Base.LinAlg.BLAS.trsm
```

```
INFO: Loading help data...

Base.LinAlg.BLAS.trsm(side, ul, tA, dA, alpha, A, B)

   Returns the solution to "A*X = alpha*B" or one of the other three
   variants determined by "side" (A on left or right of "X") and
   "tA" (transpose A). Only the "ul" triangle of "A" is used.
   "dA" indicates if "A" is unit-triangular (the diagonal is
   assumed to be all ones).
```

In [8]:
```
Base.LinAlg.BLAS.trsm('L', 'L', 'N', 'N', 1.0, full(lw), covar)
```

Out[8]:
```
5x5 Array{Float64,2}:
 1.0   0.5          0.5          0.5       0.5
 0.0   0.866025     0.288675     0.288675  0.288675
 0.0  -6.7987e-17   0.816497     0.204124  0.204124
 0.0  -5.26625e-17  3.51083e-17  0.790569  0.158114
 0.0   0.0          0.0          0.0       0.774597
```

In [9]:
```
\(full(lw), covar)
```

Out[9]:
```
5x5 Array{Float64,2}:
 1.0   0.5          0.5          0.5       0.5
 0.0   0.866025     0.288675     0.288675  0.288675
 0.0  -6.7987e-17   0.816497     0.204124  0.204124
 0.0  -5.26625e-17  3.51083e-17  0.790569  0.158114
 0.0   0.0          0.0          0.0       0.774597
```

In [10]:
```
?Base.LinAlg.BLAS.trsv
```

```
Base.LinAlg.BLAS.trsv(side, ul, tA, dA, alpha, A, b)

   Returns the solution to "A*X = alpha*b" or one of the other three
   variants determined by "side" (A on left or right of "X") and
   "tA" (transpose A). Only the "ul" triangle of "A" is used.
   "dA" indicates if "A" is unit-triangular (the diagonal is
   assumed to be all ones).
```

In [11]:
```
Base.LinAlg.BLAS.trsv('L','L', 'N', 'N', 1.0, full(lw), meanVec)
```

```
`trsv` has no method matching trsv(::Char, ::Char, ::Char, ::Char, ::Floa
t64, ::Array{Float64,2}, ::Array{Float64,1})
while loading In[11], in expression starting on line 1
```

In [12]:
```
Base.LinAlg.BLAS.trsv('L', 'N', 'N', full(lw), meanVec)
```

Out[12]:
```
5-element Array{Float64,1}:
  0.92027
 -0.0934193
  0.0505053
  0.534658
 -0.28751
```

In [13]:
```
\(full(lw), meanVec)
```

Out[13]:
```
5-element Array{Float64,1}:
  0.92027
 -0.0934193
  0.0505053
  0.534658
 -0.28751
```

In [14]: