In [76]:

```
using JuMP
using DataFrames
```

In [77]:

```
matrix = readtable("corr_matrix.csv")
```

Out[77]:

	US	UK	France	Germany	Japan	RETURN	DEVIATION
1	1.0	0.5003	0.4398	0.3681	0.2663	0.1355	0.1535
2	0.5003	1.0	0.542	0.4265	0.3581	0.1589	0.243
3	0.4398	0.542	1.0	0.6032	0.3923	0.1519	0.2324
4	0.3681	0.4265	0.6032	1.0	0.3663	0.1435	0.2038
5	0.2663	0.3581	0.3923	0.3663	1.0	0.1497	0.2298

In [79]:

```
countries = ["US", "UK", "France", "Germany", "Japan"]
#risk-free rate
rfrate = 0.05
```

Out[79]:

```
0.05
```

In [80]:

```
m = Model()
```

Out[80]:

$$\min \quad 0$$

Subject to

In [94]:

```
@variable(m, 1>= x[1:10] >= -1)
```

Out[94]:

$$-1 \leq x_i \leq 1 \quad \forall i \in \{1, 2, \dots, 9, 10\}$$

In [95]:

```
# Portfolio weights
# The first five variables for the min var portfolio, the second five for the efficient por
@constraint(m, sum{x[i], i in 1:5} == 1)
@constraint(m, sum{x[i], i in 6:10} == 1)
```

Out[95]:

$$x_6 + x_7 + x_8 + x_9 + x_{10} = 1$$

In [96]:

```
m2 = zeros(5,5)
for i in 1:5
    for j in 1:5
        if j == i
            #stdev **2
            m2[i,j] = matrix[i,7]^2
        end
        if i != j
            #correlation * stdev1 * stdev2
            m2[i, j] = matrix[i,j]*matrix[i,7]*matrix[j,7]
        end
    end
end
```

In [97]:

```
m2
```

Out[97]:

```
5x5 Array{Float64,2}:
 0.0235622   0.0186614  0.0156892  0.0115154  0.00939355
 0.0186614   0.059049   0.0306085  0.0211217  0.0199968
 0.0156892   0.0306085  0.0540098  0.0285694  0.020951
 0.0115154   0.0211217  0.0285694  0.0415344  0.017155
 0.00939355  0.0199968  0.020951   0.017155   0.052808
```

In [98]:

```
#solve for minimum variance
#all we are left to do is to multiply by weight1 and weight 2 (our variables).
#Same values if i = j
@objective(m, Min, sum{m2[i,j]*x[i]*x[j], i in 1:5, j in 1:5})
```

Out[98]:

```
:Min
```

In [99]:

```
solve(m)
```

Out[99]:

```
:Optimal
```

In [100]:

```
objvals = getvalue(x)
for i in 1:5
    println(countries[i], ":", objvals[i])
end
```

```
US:0.6345040584374309
UK:-0.01629155730330616
France:-0.01522084299990288
Germany:0.21559207328896868
Japan:0.18141626857680965
```

In [105]:

```
#Now, let's solve it for the efficient portfolio
#Maximize Sharpe ratio = (portfolio return - risk free rate)/stdev
#portfolio return = sum{c[i,1]*y[i]}
#risk free rate = constant >0 , so can be dropped altogether?P
#stdev. We have constants in m2 and we need them multiplied by the two weights
#Use the second half of variables - x[6] through x[10]
@objective(m, Max, (sum{c[i-5,1]*x[i], i in 6:10} - rfrate )/
(sum{m2[i-5,j-5]*x[i]*x[j], i in 6:10, j in 6:10}^0.5))
```

```
UndefVarError: j not defined

 [inlined code] from C:\Users\User\.julia\v0.4\JuMP\src\parseExpr_staged.jl:
340
 in anonymous at C:\Users\User\.julia\v0.4\JuMP\src\macros.jl:670
```

In [106]:

```
#probably no j in the left part so let's work around it
jays = [0.2, 0.2, 0.2, 0.2, 0.2]
@objective(m, Max, (sum{c[i-5,1]*x[i]*jays[j-5], i in 6:10, j in 6:10} - rfrate) /
(sum{m2[i-5,j-5]*x[i]*x[j], i in 6:10, j in 6:10}^0.5))
```

```
TypeError: Type{...} expression: expected Type{T}, got Function

 [inlined code] from C:\Users\User\.julia\v0.4\JuMP\src\parseExpr_staged.jl:
340
 in anonymous at C:\Users\User\.julia\v0.4\JuMP\src\macros.jl:670
```

In [ ]:

```
#still doesn't work:(
```