In [28]:
```
using JuMP, Gurobi, DataFrames
```

In [29]:
```
Inv = readtable("Investment.csv")
```

Out[29]:

	Number	Stock	Number_of_Shares	Price_Purchased_Last_Year	Current_Price	Next_Year_Price_Estimate
1	1	Yahoo!	150	15.68	31.8	29.5
2	2	General Electric	150	22.1	24.28	26.31
3	3	Microsoft	150	30.39	32.5	34.55
4	4	Bank of America	150	8.93	14.16	15.23
5	5	JPMorgan Chase	150	40.55	50.99	62.43
6	6	Cisco Systems, Inc	150	18.58	24.17	26.68
7	7	Intel	150	22.54	23.67	23.85
8	8	Pfizer	150	24.84	28.77	31.66

In [30]:
```
n = 1:8
```

Out[30]:
```
1:8
```

In [36]:
```
m = Model(solver=GurobiSolver())
```

Out[36]:
$$\begin{aligned} & \min \quad 0 \\ & \text{Subject to} \end{aligned}$$

In [37]:
```
@variable(m, 0 <= x[n] <= 150)
```

Out[37]:
$$0 \leq x_i \leq 150 \quad \forall i \in \{1, 2, \ldots, 7, 8\}$$

In [38]:
```
@constraint(m, sum((x[i]*Inv[i,5])-(0.01*x[i]*Inv[i,5])-(0.3*(x[i]*(Inv[i,5]-Inv[i,4]))) for i = n)>= 10000)
```

Out[38]:
$$\begin{aligned} 26.646x_1 + 23.383200000000002x_2 + 31.541999999999994x_3 + 12.449399999999999x_4 + 47.348099999999995x_5 + 22.2512 \\ + 23.094300000000004x_7 + 27.3033x_8 \geq 10000 \end{aligned}$$

In [39]:
```
@objective(m, Max, sum(Inv[i,6]*(150-x[i]) for i = n))
```

Out[39]: $-29.5x_1 - 26.31x_2 - 34.55x_3 - 15.23x_4 - 62.43x_5 - 26.68x_6 - 23.85x_7 - 31.66x_8 + 37531.5$

In [43]:
```
solve(m)
```

Out[43]:
```
:Optimal

Optimize a model with 1 rows, 8 columns and 8 nonzeros
Coefficient statistics:
  Matrix range     [1e+01, 5e+01]
  Objective range  [2e+01, 6e+01]
  Bounds range     [2e+02, 2e+02]
  RHS range        [1e+04, 1e+04]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -1.0757837e+04   0.000000e+00   0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective -1.075783729e+04
```

In [44]:
```
getobjectivevalue(m)
```

Out[44]:
```
26773.662707348194
```

In [45]:
```
getvalue(x)
```

Out[45]:
```
x: 1 dimensions:
[1] = 67.72329805599341
[2] = 0.0
[3] = 150.0
[4] = 0.0
[5] = 0.0
[6] = 0.0
[7] = 150.0
[8] = 0.0
```

In [47]:
```
@constraint(m, Ub[i = n], x[i] <= 75)
```

Out[47]:
```
JuMP.JuMPArray{JuMP.ConstraintRef,1,Tuple{UnitRange{Int64}}}(JuMP.ConstraintRef[x[1] <= 75, x[2] <= 75, x[3] <
= 75, x[4] <= 75, x[5] <= 75, x[6] <= 75, x[7] <= 75, x[8] <= 75], (1:8,), (Dict{Int64,Int64}(),), Dict{Symbo
l,Any}())
```

In [48]:
```
solve(m)
```

Out[48]:
```
:Optimal

Optimize a model with 9 rows, 8 columns and 16 nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 5e+01]
  Objective range  [2e+01, 6e+01]
  Bounds range     [2e+02, 2e+02]
  RHS range        [8e+01, 1e+04]
Iteration    Objective       Primal Inf.    Dual Inf.      Time
       0   -1.0757837e+04   1.500000e+02   0.000000e+00      0s
       5   -1.1062959e+04   0.000000e+00   0.000000e+00      0s

Solved in 5 iterations and 0.00 seconds
Optimal objective -1.106295884e+04
```

In [49]:
```
getobjectivevalue(m)
```

Out[49]:
```
26468.541160516463
```

In [50]:
```
getvalue(x)
```

Out[50]:
```
x: 1 dimensions:
[1] = 75.0
[2] = 75.0
[3] = 75.0
[4] = 0.0
[5] = 0.0
[6] = 4.599281839712753
[7] = 75.0
[8] = 75.0
```

In [ ]:
```

```