Vaccine Production

Vaccine production is a classic **mixing** production problem. Given raw-materials in 16 bags, production is performed in 4 weeks. In this problem we assume a simple linear relation between the mixed finished product and the raw products.

Problem

- Maximize the number of vials, doses, produced.

Sets

- $b \in Bags = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16\}$
- $w \in Weeks = \{1, 2, 3, 4\}$

Parameters

- $Vials_{b,w}$: Number of vials, doses, available in bag b in week w
- $Buffer_{b,w}$: Buffer value for each bag b in each week w

Decision variables

- Amount of raw material from bag b used in week w: $x_{b,w} \geq 0$

Model

Objective:

- Maximize the number of vaccine doses, subject to constraints:

$$\sum_{b,w} Vials_{b,w} \cdot x_{b,w}$$

Constraints:

- The material in each bag b can only be used once:

$$\sum_{w} x_{b,w} \leq 1 \ \forall \ b$$

- Maximum dose production level pr. week w:

$$\sum_{b} Vials_{b,w} \cdot x_{b,w} \leq 120000 \ \forall \ w$$

- Minimum dose production level pr. week w:

$$\sum_{b} x_{b,w} \geq 60000 \ \forall \ w$$

- Ensure correct buffer level for each week w:

$$\sum_{b} Buffer_{b,w} \cdot x_{b,w} \geq 0 \forall w$$

In the above model we simply assume that the finished product is equal to the sum of materials.

The full model in Julia/JuMP, available with the name

`VaccineProduction.jl`

from the book web-site, is given below:

```
#************************************************************************
# Vaccine Production assignment, LP
using JuMP
using HiGHS
#************************************************************************

#************************************************************************
# Data
Bags=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16]
B=length(Bags)
Weeks=[1 2 3 4]
W=length(Weeks)

Vials = [29317 29223 29129 29035;
         18551 18491 18432 18373;
```

```
         29441 29346 29252 29158;
         16971 16916 16862 16808;
         37084 36965 36846 36727;
         19288 19226 19164 19102;
         16315 16262 16210 16158;
         10949 10914 10879 10844;
         14071 14025 13980 13935;
         18002 17945 17887 17829;
         10766 10731 10697 10662;
         18190 18131 18073 18015;
         18296 18237 18178 18120;
         24426 24347 24269 24191;
         19768 19704 19641 19577;
         19214 19153 19091 19030]

Buffer = [ 6.26 6.21 6.17 6.12;
           0.23 0.20 0.17 0.14;
           6.29 6.24 6.19 6.15;
           0.21 0.19 0.16 0.13;
           7.92 7.86 7.80 7.74;
           0.24 0.21 0.18 0.15;
           -2.14 -2.17 -2.19 -2.22;
           -3.59 -3.61 -3.63 -3.65;
           -1.50 -1.52 -1.55 -1.57;
           -2.37 -2.39 -2.42 -2.45;
           -3.53 -3.55 -3.57 -3.58;
           -1.94 -1.97 -2.00 -2.03;
           -2.40 -2.43 -2.46 -2.49;
            3.57 3.54 3.50 3.46;
           -2.11 -2.14 -2.17 -2.2;
           -0.43 -0.46 -0.49 -0.52]
#*************************************************************************

#*************************************************************************
# Model
Vaccine = Model(HiGHS.Optimizer)

#proportion of bag b processed in week w
@variable(Vaccine, x[1:B,1:W] >= 0)

# Objective Function - Maximize the number of vials produced
@objective(Vaccine, Max, sum(Vials[b,w] * x[b,w] for b=1:B, w=1:W))

# Each bag can be used at most once
@constraint(Vaccine, [b=1:B], sum(x[b,w] for w=1:W) <= 1)
# At most 120,000 vials can be produced each week
```

```julia
@constraint(Vaccine, [w=1:W], sum(Vials[b,w] * x[b,w] for b=1:B) <= 120000)
# At least 60,000 vials must be produced each week
@constraint(Vaccine, [w=1:W], sum(Vials[b,w] * x[b,w] for b=1:B) >= 60000)
# The combined buffer volume needed each week must be positive
@constraint(Vaccine, [w=1:W], sum(Buffer[b,w] * x[b,w] for b=1:B) >= 0)

optimize!(Vaccine)

if termination_status(Vaccine) == MOI.OPTIMAL
    println("Optimal objective value (vial count): ", objective_value(Vaccine))
    for w=1:W
        println("Week: ",w)
        for b=1:B
            if value(x[b,w])>0.001
                println("    Usage of batch: ", b, "  amount: ", value(x[b,w]))
            end
        end
        println("")
    end
else
    println("Problem not solved to optimality")
end
#************************************************************************
```

Notice that to clean up the printout of the solution, only non-zero production usage is printed.