CS524 - Homework 1

In [351…

```
# Check JuMP version
Pkg.status("JuMP")

# Install optimizers
using Pkg
Pkg.add("SCS")
Pkg.add("ECOS")
Pkg.add("Clp")
Pkg.add("LinearAlgebra")
```

```
      Status `~/.julia/environments/v1.7/Project.toml`
  [4076af6c] JuMP v0.22.2
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.7/Project.toml`
  No Changes to `~/.julia/environments/v1.7/Manifest.toml`
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.7/Project.toml`
  No Changes to `~/.julia/environments/v1.7/Manifest.toml`
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.7/Project.toml`
  No Changes to `~/.julia/environments/v1.7/Manifest.toml`
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.7/Project.toml`
  No Changes to `~/.julia/environments/v1.7/Manifest.toml`
```

QUESTION 1a: Clp vs. ECOS vs. SCS

Clp gives an integer solution (1, 3, 3) whereas ECOS and SCS return a floating point solution (1.7, 3, 1.6)
Both are optimal solutions but Clp has exact values which I believe is better due to less variation.
Also Clp had the fastest runtime.

QUESTION 1b: Model using Clp

In [352…

```
using JuMP, Clp, ECOS, SCS, LinearAlgebra

# obj: max 6x - 2y + 3z
# con: 2x - y + z <= 2
#      0 <= x <= 3    0 <= y <= 3    0 <= z <= 3
A = [2 -1 1; 1 0 0; 0 1 0; 0 0 1];
b = [2; 3; 3; 3];

m = Model(with_optimizer(Clp.Optimizer))

@variable(m, x[1:3]>=0) # init x variables
@objective(m, Max, 6*x[1] - 2*x[2] + 3*x[3])
for i = 1:size(A,1)
    @constraint(m, A[i,:]'*x <= b[i])
end

println(m)

println("Using Clp optimizer")
optimize!(m)
println("Solution is: ", termination_status(m))
println("The optimal solution is: ", objective_value(m))
println("The values are: ", value.(x))

# println("Using ECOS optimizer")
# set_optimizer(m, ECOS.Optimizer)
# @time optimize!(m);
# println("Solution is: ", termination_status(m))
# println("The optimal solution is: ", objective_value(m))
# println("The values are: ", value.(x))

# println("Using SCS optimizer")
# set_optimizer(m, SCS.Optimizer)
# @time optimize!(m);
# println("Solution is: ", termination_status(m))
# println("The optimal solution is: ", objective_value(m))
# println("The values are: ", value.(x))
```

```
Max 6 x[1] - 2 x[2] + 3 x[3]
Subject to
 2 x[1] - x[2] + x[3] ≤ 2.0
 x[1] ≤ 3.0
 x[2] ≤ 3.0
 x[3] ≤ 3.0
 x[1] ≥ 0.0
 x[2] ≥ 0.0
 x[3] ≥ 0.0

Using Clp optimizer
Solution is: OPTIMAL
The optimal solution is: 9.0
The values are: [1.0, 3.0, 3.0]
Coin0506I Presolve 1 (-3) rows, 3 (0) columns and 3 (-3) elements
Clp0006I 0  Obj -0 Dual inf 8.999998 (2)
Clp0006I 1  Obj 9
Clp0000I Optimal - objective value 9
Coin0511I After Postsolve, objective 9, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 9 - 1 iterations time 0.002, Presolve 0.00
```

QUESTION 1c: Optimal solution

The optimal objective value is 9 and x1 = 1, x2 = 3, x3 = 3

QUESTION 1d: Infeasible solution

In [353…

```
bad_m = Model(with_optimizer(Clp.Optimizer));

@variable(bad_m, x[1:3]>=0);

for i = 1:size(A,1)
    @constraint(bad_m, A[i,:]'*x >= b[i]); # change to >= to cause infeasible (used to be <=)
end

@objective(bad_m, Max, 6*x[1] - 2*x[2] + 3*x[3]);

print(bad_m)
println()
optimize!(bad_m);
println("Solution is: ", termination_status(bad_m)) # infeasible or unbounded
```

$$\begin{aligned} \max \quad & 6x_1 - 2x_2 + 3x_3 \\ \text{Subject to} \quad & 2x_1 - x_2 + x_3 \geq 2.0 \\ & x_1 \geq 3.0 \\ & x_2 \geq 3.0 \\ & x_3 \geq 3.0 \\ & x_1 \geq 0.0 \\ & x_2 \geq 0.0 \\ & x_3 \geq 0.0 \end{aligned}$$

```
Solution is: DUAL_INFEASIBLE
Coin0508I Presolve thinks problem is unbounded
Clp3003W Analysis indicates model infeasible or unbounded
Clp0006I 0  Obj 0 Primal inf 11 (4) Dual inf 8.9999998 (2)
Clp0006I 1  Obj 9e+10
Clp0006I 1  Obj 4.5e+11
Clp0002I Dual infeasible - objective value 4.5e+11
Clp0032I DualInfeasible objective 4.5e+11 - 1 iterations time 0.002
```

QUESTION 2a: Formulate Stigler optimization problem

Decision variables:

- food type (77 types)
- nutrient (calories, protein, calcium, iron, vitamin A, thiamine, riboflavin, niacin, ascorbic acid)

Objective:

- minimize diet cost while meeting the recommended nutrient amount

Constraints:

- RDA of the 9 nutrients

QUESTION 2b: Implementation

In [355…

```
using CSV
using DataFrames
using JuMP, Clp, LinearAlgebra

raw = CSV.read("stigler.csv", DataFrame) # import Stigler's data set as a dataframe
(m,n) = size(raw)                        # m = number of rows, n = number of columns
n_nutrients = 2:n                        # indices of columns containing nutrients (skip the first one)
n_foods = 2:m                            # indices of rows containing food names (skip the two first ones)
nutrients = names(raw)[n_nutrients]      # the list of nutrients
foods = raw[n_foods,1]                   # the list of foods
data = Matrix(raw[n_foods,n_nutrients])  # put the data about nutrients and foods into an array
lower = Vector(raw[1,n_nutrients])       # lower[i] is the minimum daily requirement of nutrient i.

stigler = Model(with_optimizer(Clp.Optimizer))

@variable(stigler, f[1:length(foods)]>=0)
@objective(stigler, Min, sum(f))
for i in 1:length(nutrients)
    @constraint(stigler, data[:,i]'*f >= lower[i])
end

optimize!(stigler);
solution = objective_value(stigler)
values = value.(f)
println("Solution status: ", termination_status(stigler))
println("The diet cost per day = ", solution)
println("The diet cost per year = ", solution*365)
println("The amount of each food: ")
for i in 1:length(foods)
    if values[i] != 0
        println("    ", foods[i], " -> ", values[i])
    end
end
println()
```

```
Solution status: OPTIMAL
The diet cost per day = 0.10866227820675685
The diet cost per year = 39.66173154546625
The amount of each food:
    Wheat Flour (Enriched) -> 0.02951906167648827
    Liver (Beef) -> 0.0018925572907052643
    Cabbage -> 0.011214435246144865
    Spinach -> 0.005007660466725203
    Navy Beans, Dried -> 0.061028563526693246

Coin0506I Presolve 9 (0) rows, 76 (-1) columns and 569 (-1) elements
Clp0006I 0  Obj 0 Primal inf 5.1310537 (9)
Clp0006I 6  Obj 0.10866228
Clp0000I Optimal - objective value 0.10866228
Coin0511I After Postsolve, objective 0.10866228, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 0.1086622782 - 6 iterations time 0.002, Presolve 0.00
```

QUESTION 2c: Analysis

An optimal solution was possible.
The solution to the model I created was not super far off from Stigler's guess.
Although, my diet cost is lower by 0.27 at $39.66

My diet consisted of:

- Wheat Flour (Enriched)
- Liver (Beef)
- Cabbage
- Spinach
- Navy Beans, Dried

QUESTION 2d: Vegetarian diet

By having a vegetarian only diet, the minimum cost per year is higher at $39.80
This is expected because an alternative needs to be found to satisfy protein intake.
And this alternative (in this case its evaporated milk) costs more.

In [356…

```
# The following are meats that should be excluded:
#=
    Sirloin Steak           25
    Round Steak             26
    Rib Roast               27
    Chuck Roast             28
    Plate                   29
    Liver (Beef)            30
    Leg of Lamb             31
    Lamb Chops (Rib)        32
    Pork Chops              33
    Pork Loin Roast         34
    Bacon                   35
    Ham, smoked             36
    Salt Pork               37
    Roasting Chicken        38
    Veal Cutlets            39
    Salmon, Pink (can)      40
    Pork and Beans (can)    59
=#
# Create a new data set from the original data but without the meats listed above
vege_data = data[1:end .!= 25,:] # removes entry 25
vege_foods = foods[1:end .!= 25]
for i in 1:15 # removes other meat entries
    vege_data = vege_data[1:end .!= 25,:]
    vege_foods = vege_foods[1:end .!= 25]
end
vege_data = vege_data[1:end .!= 43,:] # removes final meat entry
vege_foods = vege_foods[1:end .!= 43]

veg = Model(with_optimizer(Clp.Optimizer))

@variable(veg, f[1:length(vege_foods)]>=0)
@objective(veg, Min, sum(f)*365)
for i in 1:length(nutrients)
    @constraint(veg, vege_data[:,i]'*f >= lower[i])
end

optimize!(veg);
solution = objective_value(veg)
values = value.(f)
println("Solution status: ", termination_status(veg))
println("The diet cost per year = ", solution)
println("The amount of each food: ")
for i in 1:length(vege_foods)
    if values[i] != 0
        println("    ", vege_foods[i], " -> ", values[i])
    end
end
```

```
Solution status: OPTIMAL
The diet cost per year = 39.79866435040896
The amount of each food:
    Wheat Flour (Enriched) -> 0.035455581408887715
    Evaporated Milk (can) -> 0.008591461668763569
    Cabbage -> 0.011249517312443502
    Spinach -> 0.005112832613199645
    Navy Beans, Dried -> 0.04862804357316849
Coin0506I Presolve 9 (0) rows, 59 (-1) columns and 446 (-1) elements
Clp0006I 0  Obj 0 Primal inf 5.2797523 (9)
Clp0006I 7  Obj 39.798664
Clp0000I Optimal - objective value 39.798664
Coin0511I After Postsolve, objective 39.798664, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 39.79866435 - 7 iterations time 0.002, Presolve 0.00
```

QUESTION 3a: Transform LP

z_1 is unbounded so we use trick 5:

$$u \geq 0, v \geq 0 \text{ and } z_1 = u - v$$

z_2, z_3, z_4 need to be nonnegative so we use trick 7:

$$0 \leq t \leq 6 \text{ and } z_2 = t - 1$$
$$0 \leq r \leq 6 \text{ and } z_3 = r - 1$$
$$0 \leq w \leq 2 \text{ and } z_4 = w - 2$$

$-z_1 + 6z_2 - z_3 + z_4 \geq -3$ needs to be an equality so we use trick 4:

$$(u - v) - 6(t - 1) + (r - 1) - (w - 2) \leq 3$$
$$u - v - 6t + 6 + r - 1 - w + 2 - 3 \leq 0$$
$$u - v - 6t + r - w + s = -4 \text{ and } s \geq 0$$

$z_3 + z_4 \leq 2$ needs to be an equality so we use trick 4:

$$r - 1 + w - 2 \leq 2$$
$$r + w - 5 \leq 0$$
$$r + w + q = 5 \text{ and } q \geq 0$$

$7z_2 + z_4 = 5$ needs to be using the new variables so we just substitute:

$$7(t - 1) + (w - 2) = 5$$
$$7t - 7 + w - 2 = 5$$
$$7t + w = 14$$

maximize $3z_1 - z_2$ needs to converted to a minimize so we use trick 1:

$$\begin{aligned} \text{maximize} \quad & 3(u - v) - (t - 1) \\ \text{maximize} \quad & 3u - 3v - t + 1 \\ \text{minimize} \quad & -3u + 3v + t - 1 \\ \text{minimize} \quad & -3u + 3v + t \end{aligned}$$

New LP:

$$\begin{aligned} \text{minimize} \quad & -3u + 3v + t \\ \text{subject to:} \quad & u - v - 6t + r - w + s = -4 \\ & 7t + w = 14 \\ & r + w + q = 5 \end{aligned}$$

$$u, v, t, r, w, s, q \geq 0$$
$$t, r \leq 6$$
$$w \leq 2$$

$$\begin{aligned} \text{where:} \quad & z_1 = u - v \\ & z_2 = t - 1 \\ & z_3 = r - 1 \\ & z_4 = w - 1 \\ & \text{original cost} = -(\text{new cost}) + 1 \end{aligned}$$

In [357…

```
# Vector values: A, x, b, c
println("                                        [ u ]")
println("                                        [ v ]")
println("    [ 1 -1 -6  1 -1  1  0 ]        [ t ]        [ -4 ]")
println("A = [ 0  0  7  0  1  0  0 ]    x = [ r ]    b = [ 14 ]    c = [-3 3 1 0 0 0 0]")
println("    [ 0  0  0  1  1  0  1 ]        [ w ]        [  5 ]")
println("                                        [ s ]")
println("                                        [ q ]")
```

```
                                   [ u ]
                                   [ v ]
    [ 1 -1 -6  1 -1  1  0 ]        [ t ]        [ -4 ]
A = [ 0  0  7  0  1  0  0 ]    x = [ r ]    b = [ 14 ]    c = [-3 3 1 0 0 0 0]
    [ 0  0  0  1  1  0  1 ]        [ w ]        [  5 ]
                                   [ s ]
                                   [ q ]
```

Question 3b: Modelling both LPs

In [1]:

```
using JuMP, Clp, LinearAlgebra

# original LP model
m_orig = Model(with_optimizer(Clp.Optimizer))

@variable(m_orig, z1)
@variable(m_orig, -1 <= z2 <= 5)
@variable(m_orig, -1 <= z3 <= 5)
@variable(m_orig, -2 <= z4 <= 2)
@objective(m_orig, Max, 3*z1 - z2)
@constraint(m_orig, -z1 + 6*z2 - z3 + z4 >= -3)
@constraint(m_orig, 7*z2 + z4 == 5)
@constraint(m_orig, z3 + z4 <= 2)

optimize!(m_orig)
println("ORIGINAL LP MODEL")
println("    Solution is: ", termination_status(m_orig))
println("    The optimal solution is: ", objective_value(m_orig))
println("    The values are: ")
println("            z1 = ", value.(z1), "    z2 = ", value.(z2), "    z3 = ", value.(z3), "    z4 = ", value.(z4))
println()
```

```
ORIGINAL LP MODEL
    Solution is: OPTIMAL
    The optimal solution is: 25.28571428571429
    The values are:
            z1 = 8.571428571428571    z2 = 0.42857142857142855    z3 = -1.0    z4 = 2.0

Coin0506I Presolve 0 (-3) rows, 0 (-4) columns and 0 (-8) elements
Clp3002W Empty problem - 0 rows, 0 columns and 0 elements
Clp0000I Optimal - objective value 25.285714
Coin0511I After Postsolve, objective 25.285714, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective 25.28571429 - 0 iterations time 0.002, Presolve 0.00
```

In [2]:

```
using JuMP, Clp, LinearAlgebra

# transformed LP model
m_alt = Model(with_optimizer(Clp.Optimizer))

A = [1 -1 -6  1 -1  1  0; 0  0  7  0  1  0  0; 0  0  0  1  1  0  1]
b = [-4; 14; 5]
c = [-3 3 1 0 0 0 0]
c = c'

@variable(m_alt, x[1:7]>=0)
set_upper_bound(x[3], 6)
set_upper_bound(x[4], 6)
set_upper_bound(x[5], 4)
@objective(m_alt, Min, sum(c[i]*x[i] for i=1:size(c,1)))
for i in 1:size(A,1)
    @constraint(m_alt, A[i,:]'*x == b[i])
end

optimize!(m_alt)
println("TRANSFORMED LP MODEL")
println("    Solution is: ", termination_status(m_alt))
println("    The optimal solution is: ", objective_value(m_alt))
println("    The values are: ")
values = value.(x)
for i in 1:length(values)
    println("        ", x[i], " = ", values[i])
end
println("    When coverted into the original variables we get:")
println("                      z1 = ", values[1]-values[2])
println("                      z2 = ", values[3]-1)
println("                      z3 = ", values[4]-1)
println("                      z4 = ", values[5]-2)
println("        optimal solution = ", -objective_value(m_alt) + 1)
println("    This matches the above.")
println()
```

```
TRANSFORMED LP MODEL
    Solution is: OPTIMAL
    The optimal solution is: -24.28571428571429
    The values are:
        x[1] = 8.571428571428571
        x[2] = 0.0
        x[3] = 1.4285714285714286
        x[4] = 0.0
        x[5] = 4.0
        x[6] = 0.0
        x[7] = 1.0
    When coverted into the original variables we get:
                  z1 = 8.571428571428571
                  z2 = 0.4285714285714286
                  z3 = -1.0
                  z4 = 2.0
        optimal solution = 25.28571428571429
    This matches the above.

Coin0506I Presolve 0 (-3) rows, 0 (-7) columns and 0 (-11) elements
Clp3002W Empty problem - 0 rows, 0 columns and 0 elements
Clp0000I Optimal - objective value -24.285714
Coin0511I After Postsolve, objective -24.285714, infeasibilities - dual 0 (0), primal 0 (0)
Clp0032I Optimal objective -24.28571429 - 0 iterations time 0.002, Presolve 0.00
```