HW3 ETM 540

Risto Rushford

1/27/2021

```
library(huxtable)
rm(list = ls())
library(tufte)
library(magrittr, quietly = TRUE)
suppressPackageStartupMessages(library(dplyr, quietly = TRUE))
suppressPackageStartupMessages(library(ROI, quietly = TRUE))
library(gridExtra, quietly = TRUE)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(ROI, quietly = TRUE)
library(ROI.plugin.glpk, quietly = TRUE)
library(ompr, quietly = TRUE)
library(ompr.roi, quietly = TRUE)
library(knitr)
library(pander)
```

```
##
## Attaching package: 'pander'
```

```
## The following object is masked from 'package:huxtable':
##
##     wrap
```

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = FALSE)
```

Exercise 3.1: Transportation

Background: Starting Model

We are given a situation in which there are four manufacturing plants which supply four distribution centers. In the table below we have the suppliers on the left-hand side and the distributors along the top:

```
d <- c(20, 6, 8, 16, 700, 21, 7, 10, 26, 500, 8, 18, 22, 15, 500, 12, 24, 28, 5, 600, 500, 500, 500, 60

transport <- matrix(d, nrow = 5, dimnames = c(list(c("Chicago", "Beaverton", "Eugene", "Dallas", "Capac

trnsprt <- as_hux(transport, add_colnames = TRUE, add_rownames = TRUE)
trnsprt <- set_caption(trnsprt, "Transportation cost between cities")
```

```
trnsprt <- set_bold(trnsprt, 1, everywhere)
trnsprt <- set_all_borders(trnsprt)

trnsprt
```

Table 1: Transportation cost between cities

rownames	PDX	SEA	MSP	ATL	Supply
Chicago	20	21	8	12	500
Beaverton	6	7	18	24	500
Eugene	8	10	22	28	500
Dallas	16	26	15	5	600
Capacity	700	500	500	600	-

```
NSupp <- 4
NDist <- 4

DistNames <- list("PDX", "SEA", "MSP", "ATL")  # Distributor names

SuppNames <- list("Chi", "Bea", "Eug", "Dal")    # Supplier names

Cost <- matrix(c(20, 21, 8, 12,
                 6, 7, 18, 24,
                 8, 10, 22, 28,
                 16, 26, 15, 5),
               ncol=4, dimnames=list(SuppNames, DistNames))

Supply <- c(500, 500, 500, 600)
Capacity <- c(700, 500, 500, 600)

grid.table(Cost)
```

	PDX	SEA	MSP	ATL
Chi	20	6	8	16
Bea	21	7	10	26
Eug	8	18	22	15
Dal	12	24	28	5

Given this information, we can formulate a model to determine the lowest cost way to transport product to the distributors:

Max $7 \cdot Ants + 10 \cdot Bats + 5 \cdot Cats + 24 \cdot Dogs$
s.t.:
$1 \cdot Ants + 3 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 800$ (machining hours)
$3 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 900$ (assembly hours)
$2 \cdot Ants + 3 \cdot Bats + 1 \cdot Cats + 2 \cdot Dogs \leq 480$ (testing hours)
$3 \cdot Ants + 5 \cdot Bats + 2 \cdot Cats + 4 \cdot Dogs \leq 1200$ (sensors)
$2 \cdot Ants + 7 \cdot Bats + 2 \cdot Cats + 6 \cdot Dogs \leq 500$ (paint)
$Ants, Bats, Cats, Dogs \geq 0$

Model Description

The initial model given is:

Max $7 \cdot Ants + 12 \cdot Bats + 5 \cdot Cats$
s.t.:
$1 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats \leq 800$ (machining hours)
$3 \cdot Ants + 6 \cdot Bats + 2 \cdot Cats \leq 900$ (assembly hours)
$2 \cdot Ants + 2 \cdot Bats + 1 \cdot Cats \leq 480$ (testing hours)
$2 \cdot Ants + 10 \cdot Bats + 2 \cdot Cats \leq 1200$ (sensors)
$Ants, Bats, Cats, Dogs \geq 0$

$ \begin{equation} $

Building in OMPR

Which when entered into the ompr model-building package becomes:

```
model0 <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 12*Bats + 5*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 2*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 2*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 1*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 2*Cats <= 1200) %>%  # sensors
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(model0))
```

```
## [1] "optimal"
```

```
result0 <- cbind(objective_value(model0),
                 get_solution(model0, Ants),
                 get_solution(model0, Bats),
                 get_solution(model0, Cats))
colnames(result0) <- list("Profit", "Ants", "Bats", "Cats")
rownames(result0) <- list("Solution")

grid.table(result0)
```

	Profit	Ants	Bats	Cats
Solution	2225	50	0	375

Discussion of Results

Given the three variables as originally provided in the text example, we see that upon entering all of the same inputs (objective, constraints, etc) gives us the same output and thus conclude that the model is correct and reproducible. This sets the stage for us to adapt the model as in the following sections to explore the special cases where the linear program does NOT produce a simple unique solution as desired.

Special Case 1:

Infeasible Model

Now I've been tasked with examining a case where the model is structured such that no feasible solution is produced. In the text we see an example of this type of scenario in which a sales manager signs a contract for us to produce more Ants than we have the capacity to produce. This is a real type of problem that occurs when sales and production staff are not in communication with each other. We can see similar results by setting a constraint for any of the drones to be produced at a higher quantity than there is capacity for.

Let's take the established model and look at the case where we add the constraint that $Bats \geq 250$:

Model Description

The updated model becomes

Max $7 \cdot Ants + 10 \cdot Bats + 5 \cdot Cats$
s.t.:
$1 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats \leq 800$ (machining hours)
$3 \cdot Ants + 6 \cdot Bats + 2 \cdot Cats \leq 900$ (assembly hours)
$2 \cdot Ants + 2 \cdot Bats + 4 \cdot Cats \leq 480$ (testing hours)
$2 \cdot Ants + 10 \cdot Bats + 1 \cdot Cats \leq 1200$ (sensors)
$Ants, Bats, Cats, Dogs \geq 0$
$Bats \geq 250$

Building in OMPR

Which building in **ompr* becomes

```
modelinf <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 12*Bats + 5*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 2*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 2*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 1*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 2*Cats <= 1200) %>%  # sensors
  add_constraint(Bats >= 250)                       %>%   # Infeasible constraint

  solve_model(with_ROI(solver = "glpk"))

print(solver_status(modelinf))
```

```
## [1] "infeasible"
```

```
resultinf <- cbind(objective_value(modelinf),
                   get_solution(modelinf, Ants),
                   get_solution(modelinf, Bats),
                   get_solution(modelinf, Cats))
colnames(resultinf) <- list("Profit", "Ants", "Bats", "Cats")
rownames(resultinf) <- list("Solution")

grid.table(resultinf)
```

	Profit	Ants	Bats	Cats
Solution	1440	0	120	0

Discussion of Results

Here we see that the model returns that it is infeasible to produce 250 Bats, which could be guessed by looking at the formulation of the resource constraints. We see when examining the testing hours constraint that the highest number of hours allocatable to Bats is $2 \cdot 240 = 480$ hours. Notice how the solver maxes out Bats at quantity 240.

Special Case 2:

Exploring with New Constraint

Now we explore the case where we have multiple optima. As we see in the text, this occurs when there are multiple ways to reach the same maximum profit. Let's examine the case where Cats consumes 3 times the resources as Bats but generates three times the profit.

Model Description

Max $7 \cdot Ants + 10 \cdot Bats + 30 \cdot Cats$
s.t.:
$1 \cdot Ants + 4 \cdot Bats + 12 \cdot Cats \leq 800$ (machining hours)
$3 \cdot Ants + 6 \cdot Bats + 18 \cdot Cats \leq 900$ (assembly hours)
$2 \cdot Ants + 2 \cdot Bats + 6 \cdot Cats \leq 480$ (testing hours)
$2 \cdot Ants + 10 \cdot Bats + 30 \cdot Cats \leq 1200$ (sensors)
$Ants, Bats, Cats, Dogs \geq 0$

Building in OMPR

```
modelmult <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 10*Bats + 30*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 12*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 18*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 6*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 30*Cats <= 1200) %>%  # sensors
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(modelmult))
```

```
## [1] "optimal"
```

```
resultmult <- cbind(objective_value(modelmult),
                get_solution(modelmult, Ants),
                get_solution(modelmult, Bats),
                get_solution(modelmult, Cats))
colnames(resultmult) <- list("Profit", "Ants", "Bats", "Cats")
rownames(resultmult) <- list("Solution")

grid.table(resultmult)
```

	Profit	Ants	Bats	Cats
Solution	1860	180	0	20

Discussion of Results

Here we see that zero Bats are built. However if we run it again with Cats constrained to zero we get

```
modelmult1 <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 10*Bats + 30*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 12*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 18*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 6*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 30*Cats <= 1200) %>%  # sensors
  add_constraint(Cats == 0) %>%
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(modelmult1))
```

```
## [1] "optimal"
```

```
resultmult1 <- cbind(objective_value(modelmult),
                get_solution(modelmult1, Ants),
                get_solution(modelmult1, Bats),
                get_solution(modelmult1, Cats))
colnames(resultmult1) <- list("Profit", "Ants", "Bats", "Cats")
rownames(resultmult1) <- list("Solution")

grid.table(resultmult1)
```

	Profit	Ants	Bats	Cats
Solution	1860	180	60	0

And just as in the the textbook example we see that we have reached the same profit output as if we produced zero Bats but instead allocated those resources to Cats. This happens due to what is called "Linear dependence" in linear algebra and can be avoided by ensuring that no product parameters is an exact multiple of another product's parameters.

Special Case 3:

Exploring with New Constraint

Here we'll analyze the case where we have a redundant constraint, which is hwere a constraint is added that doesn't have an effect on the feasible region. Let's examine if we add a constraint that each drone requires a 4 landing pad cushions of an abundant supply

Model Description

Max $7 \cdot Ants + 12 \cdot Bats + 5 \cdot Cats$
s.t.:
$1 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats \leq 800$ (machining hours)
$3 \cdot Ants + 6 \cdot Bats + 2 \cdot Cats \leq 900$ (assembly hours)
$2 \cdot Ants + 2 \cdot Bats + 1 \cdot Cats \leq 480$ (testing hours)
$2 \cdot Ants + 10 \cdot Bats + 2 \cdot Cats \leq 1200$ (sensors)
$4 \cdot Ants + 4 \cdot Bats + 4 \cdot Cats \leq 2200$ (pads)
$Ants, Bats, Cats, Dogs \geq 0$

Building in OMPR

```
modelred <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 12*Bats + 5*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 2*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 2*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 1*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 2*Cats <= 1200) %>%  # sensors
  add_constraint(4*Ants + 4*Bats + 4*Cats <= 2200) %>%  # pads
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(modelred))
```

```
## [1] "optimal"
```

```
resultred <- cbind(objective_value(modelred),
                get_solution(modelred, Ants),
                get_solution(modelred, Bats),
                get_solution(modelred, Cats))
colnames(resultred) <- list("Profit", "Ants", "Bats", "Cats")
rownames(resultred) <- list("Solution")

grid.table(resultred)
```

	Profit	Ants	Bats	Cats
Solution	2225	50	0	375

Discussion of Results

Notice how these results are no different than in the original base case! This is because the supply of pads puts no real constraint on the amount of product capacity given other available resources.

Special Case 4

Unfortunately, I had a misunderstanding about the requirements of this assignment and started on the wrong track before realizing that I was supposed to be replicating the **four special cases** and instead I followed Exercise 2.1 in the text. Since I ran out of time to complete Special Case 4, I will go ahead and insert what I had previously finished in case you want to see it.

Exercise 2.1: Profit Optimization with Animal-Themed Drones

Part 1

Background: Starting Model

I have been given an initial model to use for today's exercise which is composed of four animal-themed robotic drones. These drones are produced in a manufacturing process which splits resources between each drone as a product, and the goal of the model is to find the optimal production values of each drone in order to meet the objective of maximizing profit.

Model Description

The initial model given is:

Max $7 \cdot Ants + 12 \cdot Bats + 5 \cdot Cats$
s.t.:
$1 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats \leq 800$ (machining hours)
$3 \cdot Ants + 6 \cdot Bats + 2 \cdot Cats \leq 900$ (assembly hours)
$2 \cdot Ants + 2 \cdot Bats + 1 \cdot Cats \leq 480$ (testing hours)
$2 \cdot Ants + 10 \cdot Bats + 2 \cdot Cats \leq 1200$ (sensors)
$Ants, Bats, Cats, Dogs \geq 0$

$ \begin{equation} $

Building in OMPR

Which when entered into the ompr model-building package becomes:

```
modela <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "continuous", lb = 0) %>%
  add_variable(Bats, type = "continuous", lb = 0) %>%
  add_variable(Cats, type = "continuous", lb = 0) %>%

  set_objective(7*Ants + 12*Bats + 5*Cats, "max") %>%

  add_constraint(1*Ants + 4*Bats + 2*Cats <= 800) %>%    # machining hours
  add_constraint(3*Ants + 6*Bats + 2*Cats <= 900) %>%    # assembly hours
  add_constraint(2*Ants + 2*Bats + 1*Cats <= 480) %>%    # testing hours
  add_constraint(2*Ants + 10*Bats + 2*Cats <= 1200) %>%  # sensors
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(modela))
```

```
## [1] "optimal"
```

```
resulta <- cbind(objective_value(modela),
                 get_solution(modela, Ants),
                 get_solution(modela, Bats),
                 get_solution(modela, Cats))
colnames(resulta) <- list("Profit", "Ants", "Bats", "Cats")
rownames(resulta) <- list("Solution")

grid.table(resulta)
```

	Profit	Ants	Bats	Cats
Solution	2225	50	0	375

Discussion of Results

Given the three variables as originally provided in the text example, we see that upon entering all of the same inputs (objective, constraints, etc) gives us the same output and thus conclude that the model is correct and reproducible. This sets the stage for us to adapt the model as in the next section.

Part 2

Updating the Model

I have been tasked with adding a new product to the mix. Additionally a new painting department has been added to the company, adding a new resource of paint to the mix.

Model Description

The updated model becomes

Max $7 \cdot Ants + 10 \cdot Bats + 5 \cdot Cats + 24 \cdot Dogs$
s.t.:
$1 \cdot Ants + 3 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 800$ (machining hours)
$3 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 900$ (assembly hours)
$2 \cdot Ants + 3 \cdot Bats + 1 \cdot Cats + 2 \cdot Dogs \leq 480$ (testing hours)
$3 \cdot Ants + 5 \cdot Bats + 2 \cdot Cats + 4 \cdot Dogs \leq 1200$ (sensors)
$2 \cdot Ants + 7 \cdot Bats + 2 \cdot Cats + 6 \cdot Dogs \leq 500$ (paint)
$Ants, Bats, Cats, Dogs \geq 0$

Building in OMPR

Which building in **ompr* becomes

```
update0 <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "integer", lb = 0) %>%
  add_variable(Bats, type = "integer", lb = 0) %>%
  add_variable(Cats, type = "integer", lb = 0) %>%
  add_variable(Dogs, type = "integer", lb = 0) %>%

  set_objective(7*Ants + 10*Bats + 5*Cats + 24*Dogs, "max") %>%

  add_constraint(1*Ants + 3*Bats + 2*Cats + 2*Dogs <= 800) %>%  # fab hrs
  add_constraint(3*Ants + 4*Bats + 2*Cats + 2*Dogs <= 900) %>%  # assembly hrs
  add_constraint(2*Ants + 3*Bats + 1*Cats + 2*Dogs <= 480) %>%  # machine hrs
  add_constraint(3*Ants + 5*Bats + 2*Cats + 4*Dogs <= 1200) %>% # sensors
  add_constraint(2*Ants + 7*Bats + 2*Cats + 6*Dogs <= 500) %>% # paint
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(update0))
```

```
## [1] "optimal"
```

```
res0 <- cbind(objective_value(update0),
                get_solution(update0, Ants),
                get_solution(update0, Bats),
                get_solution(update0, Cats),
                get_solution(update0, Dogs))
colnames(res0) <- list("Profit", "Ants", "Bats", "Cats", "Dogs")
rownames(res0) <- list("Solution")

grid.table(res0)
```

	Profit	Ants	Bats	Cats	Dogs
Solution	1999	1	0	0	83

Discussion of Results

Note that I set the products to be integers rather than continuous as I'm assuming that it makes little sense for the company to manufacture a portion of a drone. Given the initial parameters we see that the optimization algorithm returns a product mix that's essentially all Dogs, with one Ant and that clearly is only because there were enough resources left over from making the lost complete Dog to also make one Ant.

This merits some discussion, as it may or may not be fair to assume that my company is not interested in producing only one product. But given a goal of maximizing profit with no constraints on manufacturing the Dog drones other than the total cap on each resource, it was really a predictable result.

But what about market forces? If we produce only the one product, then it may saturate the market and either reduce the utility of the product and thus it's value to consumers, or if the company keeps the price high then the additional units built above the demand point will be left sitting in inventory. Clearly this is not desirable as it defeats the purpose of the optimization exercise.

Part 3

Exploring with New Constraint

As per part (e) on page 49 of the text, let's explore changing some of the parameters. Let's say that we have a demand forecast that estimates that 20 Dogs will be sold and we're very confident that we can sell at least that many, but not too many more. Let's set a limit of 22 and see what we get:

Model Description

Max $7 \cdot Ants + 10 \cdot Bats + 5 \cdot Cats + 24 \cdot Dogs$
s.t.:
$1 \cdot Ants + 3 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 800$ (machining hours)
$3 \cdot Ants + 4 \cdot Bats + 2 \cdot Cats + 2 \cdot Dogs \leq 900$ (assembly hours)
$2 \cdot Ants + 3 \cdot Bats + 1 \cdot Cats + 2 \cdot Dogs \leq 480$ (testing hours)
$3 \cdot Ants + 5 \cdot Bats + 2 \cdot Cats + 4 \cdot Dogs \leq 1200$ (sensors)
$2 \cdot Ants + 7 \cdot Bats + 2 \cdot Cats + 6 \cdot Dogs \leq 500$ (paint)
$Ants, Bats, Cats, Dogs \geq 0$

Building in OMPR

```
update1 <- MIPModel() %>%    # Initialized empty model
  add_variable(Ants, type = "integer", lb = 0) %>%
  add_variable(Bats, type = "integer", lb = 10) %>%
  add_variable(Cats, type = "integer", lb = 0) %>%
  add_variable(Dogs, type = "integer", lb = 0, ub = 22) %>%

  set_objective(7*Ants + 10*Bats + 5*Cats + 24*Dogs, "max") %>%

  add_constraint(1*Ants + 3*Bats + 2*Cats + 2*Dogs <= 800) %>%  # fab hrs
  add_constraint(3*Ants + 4*Bats + 2*Cats + 2*Dogs <= 900) %>%  # assembly hrs
  add_constraint(2*Ants + 3*Bats + 1*Cats + 2*Dogs <= 480) %>%  # machine hrs
  add_constraint(3*Ants + 5*Bats + 2*Cats + 4*Dogs <= 1200) %>% # sensors
  add_constraint(2*Ants + 7*Bats + 2*Cats + 6*Dogs <= 500) %>% # paint
  solve_model(with_ROI(solver = "glpk"))

print(solver_status(update1))
```

```
## [1] "optimal"
```

```
res1 <- cbind(objective_value(update1),
                get_solution(update1, Ants),
                get_solution(update1, Bats),
                get_solution(update1, Cats),
                get_solution(update1, Dogs))
colnames(res1) <- list("Profit", "Ants", "Bats", "Cats", "Dogs")
rownames(res1) <- list("Solution")

grid.table(res1)
```

	Profit	Ants	Bats	Cats	Dogs
Solution	1671	149	10	0	22

Discussion of Results

With this update we see that the optimizer has chosen to produce the maximum number of Dogs, which was expected, and then to allocate all of the remaining resources to Ants, which are the cheapest to manufacture and have the highest return on investment. We may have a similar problem of flooding the market, only now with the much cheaper Ants. Since this product is cheaper to produce and has a high profit margin, it would make sense for a company to focus on this end of the product mix if they were looking to break into a market with cheap substitutes. That is to say, there is likely much more competition from other companies producing similar products.