ETM 540 Week 1 Homework

Jordan Hilton

January 8, 2019

Problem 1.1

I installed packages quietly in the setup for this RMD, including pander, magrittr, dplyr, ROI, and ompr.

Let's start with creating table b, and then transposing it, as from the book:

```
b<-matrix(c(1,2,3,4,5,6,7,8), ncol=4, dimnames=c(list(c("Row1", "Row2")), list(c("Col1", "Col2","Col3",
d<-t(b)
pander(d, caption="Transposition of Matrix b")
```

Table 1: Transposition of Matrix b

	Row1	Row2
Col1	1	2
Col2	3	4
Col3	5	6
Col4	7	8

We can see the transposed names. Let's change the names for the matrix so that they make sense:

```
dimnames(d)<-c(list(c("Row 1", "Row 2","Row 3","Row 4")), list(c("Column 1", "Column 2")))
pander(d, caption="Fixed Names")
```

Table 2: Fixed Names

	Column 1	Column 2
Row 1	1	2
Row 2	3	4
Row 3	5	6
Row 4	7	8

Problem 1.2

To do this let's adapt the code used above to create a matrix by changing the number of columns to seven and naming appropriately:

```
demand<-matrix(c(10,15,20,25,30,35,40), ncol=7, dimnames=c("WIDGET!!!", list(c("Monday", "Tuesday","Wedr
pander(demand, caption="Weekly Demand")
```

Table 3: Weekly Demand

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
WIDGET!!!	10	15	20	25	30	35	40

Problem 1.3

Let's create each row with seq as above, and then rbind them since we're learning about rbind:

```
demandweek1<-matrix(seq(10, by=5,length.out=7), ncol=7, dimnames=c("Week 1", list(c("Monday", "Tuesday"
demandweek2<-matrix(seq(45, by=5,length.out=7), ncol=7, dimnames=c("Week 2", list(c("Monday", "Tuesday"
demandweek3<-matrix(seq(80, by=5,length.out=7), ncol=7, dimnames=c("Week 3", list(c("Monday", "Tuesday"
demandweek4<-matrix(seq(115, by=5,length.out=7), ncol=7, dimnames=c("Week 4", list(c("Monday", "Tuesday
demandmonth<-rbind(demandweek1, demandweek2,demandweek3,demandweek4)
pander(demandmonth, caption="Weekly Demand for a Month")
```

Table 4: Weekly Demand for a Month

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Week 1	10	15	20	25	30	35	40
Week 2	45	50	55	60	65	70	75
Week 3	80	85	90	95	100	105	110
Week 4	115	120	125	130	135	140	145

You can also do this in one line as follows:

```
demand2<-matrix(seq(10, by=5,length.out=28), ncol=7, byrow=TRUE, dimnames=c(list(c("Week 1","Week 2","We
pander(demand2, caption="Another Way")
```

Table 5: Another Way

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Week 1	10	15	20	25	30	35	40
Week 2	45	50	55	60	65	70	75
Week 3	80	85	90	95	100	105	110
Week 4	115	120	125	130	135	140	145

Problem 1.4

Let's use our existing week 1 for Widgets and rbind to a similar row for gadgets:

```
widgetsweek1<-matrix(seq(10, by=5,length.out=7), ncol=7, dimnames=c("WIDGETS!", list(c("Monday", "Tuesda
gadgetsweek1<-matrix(seq(20, by=3,length.out=7), ncol=7, dimnames=c("GADGETS!", list(c("Monday", "Tuesda
twoproducts<-rbind(widgetsweek1, gadgetsweek1)
pander(twoproducts, caption="Weekly Demand for a Month")
```

Table 6: Weekly Demand for a Month

	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
WIDGETS!	10	15	20	25	30	35	40
GADGETS!	20	23	26	29	32	35	38

Problem 2.1

a (Formulation)

Using the same format of formulation as in the book, let's first clearly define our choices:

- Chairs = # of Chairs to Make
- Desks = # of Desks to Make
- Tables = # of Tables to Make
- Frames = # of Frames to Make

Now let's formulate the LP:

$$
\begin{aligned}
\text{Maximize } & 20 * Chairs + 14 * Desks + 3 * Frames + 16 * Tables \\
\text{subject to } & 6 * Chairs + 2 * Desks + Frames + 4 * Tables \leq 1440 \\
& 8 * Chairs + 6 * Desks + Frames + 8 * Tables \leq 1440 \\
& 6 * Chairs + 4 * Desks + Frames + 25 * Tables \leq 2000 \\
& 7 * Chairs + 10 * Desks + 2 * Frames + 12 * Tables \leq 1000 \\
& 40 * Chairs + 25 * Desks + 5 * Frames + 16 * Tables \leq 9600 \\
& Chairs, Desks, Frames, Tables \geq 0
\end{aligned}
$$

b (Implementation)

Let's implement the model we've described above, as done in the book:

```
result <- MIPModel() %>%
  add_variable(Chairs, type = "continuous", lb = 0) %>%
  add_variable(Desks, type = "continuous",lb = 0) %>%
  add_variable(Frames, type = "continuous", lb = 0) %>%
  add_variable(Tables, type = "continuous", lb = 0) %>%

  set_objective(20*Chairs + 14*Desks + 3*Frames + 16*Tables, "max") %>%

  add_constraint(6*Chairs+2*Desks+Frames+4*Tables <= 1440) %>% #fabrication
  add_constraint(8*Chairs+6*Desks+Frames+8*Tables <= 1440) %>% #assembly
  add_constraint(6*Chairs+4*Desks+Frames+25*Tables <= 2000) %>% #machining
  add_constraint(7*Chairs+10*Desks+2*Frames+12*Tables <= 1000) %>% #painting
  add_constraint(40*Chairs+25*Desks+5*Frames+16*Tables <= 9600) %>% #wood


  solve_model(with_ROI(solver = "glpk"))
```

c (Solution)

Let's examine the outputs from solving our MIPModel, starting with the solver status:

```
print(solver_status(result))
```

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

Hooray, we've obtained an optimal solution! How much stuff should we make?

```
get_solution(result, Chairs)
```

```
##   Chairs
## 142.8571
```

```
get_solution(result, Desks)
```

```
## Desks
##     0
```

```r
get_solution(result, Frames)
```

```
## Frames
##      0
```

```r
get_solution(result, Tables)
```

```
## Tables
##      0
```

I hope you like chairs! Let's ask the important question- how much money did we make?

```r
print(objective_value(result))
```

```
## [1] 2857.143
```

c (Interpretation and discussion)

So it looks like our model is just telling us that it's not worth it to produce anything except chairs, which must mean that they use our bottleneck resource more efficiently than the other products.

d (Changing one parameter)

I think the easiest way to obtain a different result is just to tell our model that another product is suddenly much more profitable (say, set the profit on frames to 30). It would be more interesting though to identify our bottleneck resource and increase the supply of that so that our model tells us to make a mix of products. Let's see what happens when we increase the supply of painting hours:

```r
result2 <- MIPModel() %>%
  add_variable(Chairs, type = "continuous", lb = 0) %>%
  add_variable(Desks, type = "continuous",lb = 0) %>%
  add_variable(Frames, type = "continuous", lb = 0) %>%
  add_variable(Tables, type = "continuous", lb = 0) %>%

  set_objective(20*Chairs + 14*Desks + 3*Frames + 16*Tables, "max") %>%

  add_constraint(6*Chairs+2*Desks+Frames+4*Tables <= 1440) %>% #fabrication
  add_constraint(8*Chairs+6*Desks+Frames+8*Tables <= 1440) %>% #assembly
  add_constraint(6*Chairs+4*Desks+Frames+25*Tables <= 2000) %>% #machining
  add_constraint(7*Chairs+10*Desks+2*Frames+12*Tables <= 2000) %>% #painting
  add_constraint(40*Chairs+25*Desks+5*Frames+16*Tables <= 9600) %>% #wood


  solve_model(with_ROI(solver = "glpk"))
  print(solver_status(result2))
```

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

```r
  get_solution(result2, Chairs)
```

```
##   Chairs
## 97.77778
```

```r
  get_solution(result2, Desks)
```

```
## Desks
##     0
```

```r
get_solution(result2, Frames)
```

```
##   Frames
## 657.7778
```

```r
get_solution(result2, Tables)
```

```
## Tables
##      0
```

```r
print(objective_value(result2))
```

```
## [1] 3928.889
```

Well, it looks like it's not as simple as there being one bottleneck resource- if we have more painting hours, our model tells us to make a mix of frames and chairs, but still no tables or desks. I'm guessing that tables are too inefficient in machining and desks in assembly?