DSSH 6301 - HW 01 Solutions

Here are suggested solutions to the homework assignment. Note that many problems have multiple solutions, so we present here merely one example. If you did not have points taken off for a question, we considered your solution correct. If you have questions about any of these solutions or your own graded homework, please direct questions to Matt Harrigan m.harrigan@neu.edu .

Quick tip: You can set the global options in R so that the stringsAsFactors argument in functions that use it is always false, and that way you don't have to remember to include it every time (unless you have variables that are actually categorical).

```
options(stringsAsFactors=F)
```

Problem 1

Part a

```
v1 <- 2:6
v2 <- 5:9
```

Part b

```
v2 - v1
```

```
## [1] 3 3 3 3 3
```

Part c

```
v1 %*% v2
```

```
##      [,1]
## [1,]  150
```

The %*% operation returns a matrix with 1 row and 1 column if two vectors of the same length are used in the operation.

Knowing the definition of the inner product is also useful here:

```
sum(v1*v2)
```

```
## [1] 150
```

Part d

```
v3 <- v1 + v2
v3[v3 > 10] <- 0
v3
```

```
## [1] 7 9 0 0 0
```

You can also do this with an `ifelse` statement, which we will get to next week.

```
v4 <- ifelse(v1+v2 > 10, 0, v1+v2)
v4
```

```
## [1] 7 9 0 0 0
```

Problem 2

Part a

```
m1 <- matrix(1:25, nrow=5, ncol=5)
m1
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    6   11   16   21
## [2,]    2    7   12   17   22
## [3,]    3    8   13   18   23
## [4,]    4    9   14   19   24
## [5,]    5   10   15   20   25
```

Parts b-d of this question use the matrix multiplication operator **%*%**. This returns the answer you would expect when using linear algebra to multiply matrices. If you use the multiplication operator * you get element-wise multiplication.

Part b

```
m1 %*% v1
```

```
##      [,1]
## [1,]  270
## [2,]  290
## [3,]  310
## [4,]  330
## [5,]  350
```

Part c

```
v1 %*% m1
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]   70  170  270  370  470
```

Part d

```
m1 %*% t(m1)
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  855  910  965 1020 1075
## [2,]  910  970 1030 1090 1150
## [3,]  965 1030 1095 1160 1225
## [4,] 1020 1090 1160 1230 1300
## [5,] 1075 1150 1225 1300 1375
```

Problem 3

Part a

```
dates <- seq.Date(from=Sys.Date(), by=1, length.out=5)
str_dates <- as.character(dates) # Data type coercion, a double-edged sword in R
nums <- 1:5

class(dates)
```

```
## [1] "Date"
```

```
class(str_dates)
```

```
## [1] "character"
```

```
class(nums)
```

```
## [1] "integer"
```

```
# You'll see this often as a quick way to create a data frame.
# The left hand side of each '=' is a name for the new column.
# The right hand side is a vector that you've already defined.
df <- data.frame(dates=dates, str_dates=str_dates, nums=nums)
```

Part b

```r
str(df)
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ dates    : Date, format: "2015-01-21" "2015-01-22" ...
##  $ str_dates: chr  "2015-01-21" "2015-01-22" "2015-01-23" "2015-01-24" ...
##  $ nums     : int  1 2 3 4 5
```

Part c

```r
write.csv(df, file="hw1_output.csv", row.names=F)
df_read_in <- read.csv(file="hw1_output.csv")

str(df_read_in) # Note the data type change of the dates column.
```

```
## 'data.frame':    5 obs. of  3 variables:
##  $ dates    : chr  "2015-01-21" "2015-01-22" "2015-01-23" "2015-01-24" ...
##  $ str_dates: chr  "2015-01-21" "2015-01-22" "2015-01-23" "2015-01-24" ...
##  $ nums     : int  1 2 3 4 5
```

Part d

```r
df_subset <- df[c(1, 3, nrow(df)), 1:2]
df_subset
```

```
##        dates  str_dates
## 1 2015-01-21 2015-01-21
## 3 2015-01-23 2015-01-23
## 5 2015-01-25 2015-01-25
```

Part e

```r
df[df[,3] %% 2 == 0,3] <- 0  # We can do it by column number
df$nums[df$nums %% 2 == 0] <- 0 # Or better, by column name
df$nums <- ifelse(df$nums%%2 == 0, 0, df$nums) # Or less elegantly, using ifelse
df
```

```
##        dates  str_dates nums
## 1 2015-01-21 2015-01-21    1
## 2 2015-01-22 2015-01-22    0
## 3 2015-01-23 2015-01-23    3
## 4 2015-01-24 2015-01-24    0
## 5 2015-01-25 2015-01-25    5
```

The %% operator is called the modulo operator. This operation returns the remainder of a division. When X is an even number, its remainder when divided by 2 is 0.

Part f

```
lst <- list(v1=v1, v2=v2, m1=m1, df=df)
str(lst)
```

```
## List of 4
##  $ v1: int [1:5] 2 3 4 5 6
##  $ v2: int [1:5] 5 6 7 8 9
##  $ m1: int [1:5, 1:5] 1 2 3 4 5 6 7 8 9 10 ...
##  $ df:'data.frame':  5 obs. of  3 variables:
##   ..$ dates    : Date[1:5], format: "2015-01-21" ...
##   ..$ str_dates: chr [1:5] "2015-01-21" "2015-01-22" "2015-01-23" "2015-01-24" ...
##   ..$ nums     : num [1:5] 1 0 3 0 5
```

You can refer to data members by index, like here, but this is less readable then by name.

```
lst[[3]][2]
```

```
## [1] 2
```

The third item is a matrix (m1), but given only a 1-dimensional index the matrix is vectorized.

```
mat <- lst[[3]]
nelements <- nrow(mat) * ncol(mat)

mat[1:nelements]
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25
```

```
mat[1:nelements][2]
```

```
## [1] 2
```

Problem 4

Here is the latex code used to produce the equation below:

```
$$ x = \frac{-b \pm \sqrt{b^2-4ac}}{2a} $$
```

$$x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$$

Remember that you can find more latex symbols and tips online or in the cribsheet here: http://faculty.cbu.ca/srodney/ShortSymbInd.pdf .