# STAT 210
# Applied Statistics and Data Analysis:
# Problem List 1
# Solution to Problems 3 and 4

## Exercise 3

Consider the following system of equations:

$$\begin{aligned} 4x + y + 2z + -3w &= -16 \\ -3x + 3y - z + 4w &= 20 \\ -x + 2y + 5z + w &= -4 \\ 5x + 4y + 3z - w &= -10 \end{aligned}$$

(a) Create a matrix in `R` with the coefficients of the system, and a vector with the constants on the right-hand side of the equations. Call them `mat1` and `vec1`, respectively.

**Solution:**

```
(mat1 <- matrix(c(4,-3,-1,5,1,3,2,4,2,-1,5,3,-3,4,1,-1), ncol = 4))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4    1    2   -3
## [2,]   -3    3   -1    4
## [3,]   -1    2    5    1
## [4,]    5    4    3   -1
```

```
(vec1 <- c(-16,20,-4,-10))
```

```
## [1] -16  20  -4 -10
```

(b) Create a list named `list1` having as components `mat1` and `vec1`. Call these components `item1` and `item2`, respectively. Remove `mat1` and `vec1` from the working directory.

**Solution:**

```
(list1 <- list(item1 = mat1, item2 = vec1))
```

```
## $item1
##      [,1] [,2] [,3] [,4]
## [1,]    4    1    2   -3
## [2,]   -3    3   -1    4
## [3,]   -1    2    5    1
## [4,]    5    4    3   -1
##
## $item2
## [1] -16  20  -4 -10
```

```
rm(mat1, vec1)
```

(c) Find the inverse of `item1` and store it in `list1` as `item3`. Verify that you obtained the inverse.

**Solution:** We obtain the inverse using the function `solve`:

```
list1$item3 <- solve(list1$item1)
round(list1$item1 %*% list1$item3,14) # Round to 14 decimal places
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

```
round(list1$item3 %*% list1$item1,14) # Round to 14 decimal places
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    0    0    0
## [2,]    0    1    0    0
## [3,]    0    0    1    0
## [4,]    0    0    0    1
```

(d) Solve the system of equations and store the solution in `list1` as `item4`. Verify the solution.

**Solution:** We use again the function `solve`:

```
(list1$item4 <- solve(list1$item1,list1$item2)) #Solution
```

```
## [1] -1  1 -2  3
```

```
(list1$item1 %*% list1$item4) # Verification
```

```
##      [,1]
## [1,]  -16
## [2,]   20
## [3,]   -4
## [4,]  -10
```

(e) Verify that if you multiply the inverse matrix `item3` by `item2` you also get the solution.

**Solution:**

```
(list1$item3 %*% list1$item2)
```

```
##      [,1]
## [1,]   -1
## [2,]    1
## [3,]   -2
## [4,]    3
```

(f) Find the eigenvalues of `item1` and `item3` and verify that the eigenvalues of `item3` are the reciprocals of the eigenvalues of `item1`.

**Solution:** Start by using the function `eigen`, which computes eigenvalues and eigenvectors for a square matrix

```
eigen1 <- eigen(list1$item1)
eigen3 <- eigen(list1$item3)
```

Let's look at the structure of these objects to see where the eigenvalues are stored:

```
str(eigen1)
```

```
## List of 2
##  $ values : cplx [1:4] 6.461+0i 3.523+0i 0.508+2.2i ...
##  $ vectors: cplx [1:4, 1:4] 0.0907+0i 0.3311+0i 0.7645+0i ...
##  - attr(*, "class")= chr "eigen"
```

The output of `eigen` is a list with first component `values` and second component `vectors`. We use the first component. The eigenvalues are

```
eigen1$values
```

```
## [1] 6.4613551+0.000000i 3.5229553+0.000000i 0.5078448+2.199561i
## [4] 0.5078448-2.199561i
```

```
eigen3$values
```

```
## [1] 0.09965607+0.4316271i 0.09965607-0.4316271i 0.28385259+0.0000000i
## [4] 0.15476630+0.0000000i
```

We now calculate the reciprocals for the eigenvalues of `mat2`:

```
1/eigen3$values
```

```
## [1] 0.5078448-2.199561i 0.5078448+2.199561i 3.5229553+0.000000i
## [4] 6.4613551+0.000000i
```

## Exercise 4

You will need the file Human_data.txt.

(a) Read the file Human_data.txt and store it in an object called `human`. Use the function `str` to explore the structure of this data set.

(b) Using `subset`, create a new data frame with the variables `Head_size, Height_cm, Weight_kg` from `human`. Call this new data frame `human1`.

(c) Use the function `apply` twice to calculate the mean and standard deviation for each of the three variables in `human1`. Call the vectors you obtain `human.mean` and `human.sd`.

(d) Use the function `sweep` twice, first to subtract the mean for each variable to the values in `human1` and then to divide by the standard deviation. Store the result in a data frame named `human_std`.

(e) The previous procedure, i.e., subtracting the mean and dividing by the standard deviation, is known as *standardization*. The resulting columns in the `human.std` should now have mean zero and variance equal to one. Verify this using `apply`.

(f) Another way to standardize the columns of `human1` is to use the function `scale`, which standardizes vectors. Combine this function with `apply` to obtain a standardized version for `human1` and store it in a file named `human1_std`.

(g) Show that `human_std` and `human1_std` are equal.

## Solution

(a) The file has a header, so we set the argument `header` to `TRUE` when reading the file.

```
human <- read.table('Human_data.txt', header = TRUE)
str(human)
```

```
## 'data.frame':    500 obs. of  10 variables:
##  $ Index         : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Gender        : chr  "M" "F" "M" "F" ...
##  $ age           : int  22 33 46 24 37 31 38 38 21 31 ...
##  $ Ocupation     : chr  "Nothing" "Nothing" "Work" "student" ...
##  $ Head_size     : num  34.4 28 27 24.8 30.1 26.6 25.6 25.6 27.6 23.6 ...
##  $ Height_cm     : num  206 163 162 156 173 ...
##  $ Weight_kg     : num  105.3 71.3 94.7 56 103.3 ...
##  $ Salary        : int  0 0 19268 2034 14829 10586 11272 13048 2068 12326 ...
##  $ blood_type    : int  4 4 4 3 2 3 4 2 1 3 ...
##  $ Sugar_in_blood: num  95.2 83.5 92.7 95.8 114.1 ...
```

We see that the file has 500 observation of 10 variables. Two of them are characters, four are integers and the remaining four are numerical (real).

(b) We use subset with the argument `select` to extract the variables of interest.

```
human1 <- subset(human, select = c(Head_size, Height_cm, Weight_kg))
str(human1)
```

```
## 'data.frame':    500 obs. of  3 variables:
##  $ Head_size: num  34.4 28 27 24.8 30.1 26.6 25.6 25.6 27.6 23.6 ...
##  $ Height_cm: num  206 163 162 156 173 ...
##  $ Weight_kg: num  105.3 71.3 94.7 56 103.3 ...
```

(c) Now, we calculate mean and standard deviation using `apply`.

```
human_mean <- apply(human1,2,mean)
human_sd <- apply(human1,2,sd)
```

(d) With `sweep`, we subtract the mean and divide by the standard deviation.

```
human_ctr <- sweep(human1,2,human_mean)
human_std <- sweep(human_ctr,2,human_sd,'/')
```

(e) To check that the data have been normalized, we calculate mean and variance, which should be equal to 0 and 1. We use `round` for the mean to round off the output to 15 decimal places, to obtain a neater output.

```
round(apply(human_std, 2, mean),15)
```

```
## Head_size Height_cm Weight_kg
##         0         0         0
```

```
apply(human_std, 2, sd)
```

```
## Head_size Height_cm Weight_kg
##         1         1         1
```

(f) Another way to standardize the columns of `human1` is to use the function `scale`, which standardizes vectors. Combine this function with `apply` to obtain a standardized version for `human1` and store it in a file named `human1_std`.

```
human1_std <- apply(human1, 2, scale)
```

(g) To show that `human_std` and `human1_std` are equal, we can calculate the difference between the two files, entry by entry, and then sum up these differences.

```
round(sum(human1_std-human_std),11)
```

```
## [1] 0
```

Rounding the output to 11 decimals shows that the sum of differences is zero.

We can also use the function `all.equal`. The two objects must be of the same type, so we use `as.data.frame()` on `human1_std`, which is a matrix.

```
all.equal(as.data.frame(human1_std), human_std)
```

```
## [1] TRUE
```