Homework 1 [45 points]

Questions related to the homework can be posted on the Piazza forum site, but do not post answers to homework problems. We will try to respond promptly to posted questions. However, please do not send post questions after 9pm the day before the assignment is due.

You may discuss the homework problems and computing issues with other students in the class. However, you must write up your homework solution on your own. In particular, do not share your code or homework files with other students.

This homework is meant to ease everyone gently into the `R` environment, by assessing basic `R` proficiency. The homework should take at most four hours to do, though most students will probably complete it in 1-2 hours.

If you are having any difficulties starting the homework, please contact the TA immediately.

The homework is split into two parts, covering the following:

Part 1: Simple `R` commands and the `R` environment.

Part 2: Conducting (guided) exploratory data analysis on a given dataset.

We hope that you learn something from this homework, even if you already know how to use `R`.

Files to submit on CMS

Do check to see that you have submitted all these files. Instructions on what files to submit will be in blue throughout the homework.

1. Part One [25 points]
   - `dentist.r` [5]
   - `order_fn.r` [5]
   - `harmonic.r` [15]
2. Part Two [20 points]

   You may either submit this as a html file or a pdf file using R Markdown *or* TeX this up and submit it as a pdf file. In other words, submit this as:
   - `writeup.html` or `writeup.pdf` [20]

Part One - 25 points

(i) This question is taken from `Jones O. et al.`, 4.6.1. and reproduced here.

Here are the first few lines of the files `age.txt` and `teeth.txt` (the files can be found on CMS), taken from the database of a statistically minded dentist:

```
ID      Age
1       18
2       19
3       17
.       .
.       .
.       .
```

```
ID      Num Teeth
1       28
2       27
3       32
.       .
.       .
.       .
```

Create an `R` file called `dentist.r`, and within this file, write `R` code to read in each text file (assume the text file is in your current directory), and then write an amalgamated table to the file `age_teeth.txt` of the following form:

```
ID      Age      Num Teeth
1       18       28
2       19       27
3       17       32
.       .        .
.       .        .
.       .        .
```

Upload `dentist.r` to CMS.

At a minimum, someone copying and pasting the entire contents of `dentist.r` *should get the output* `age_teeth.txt` *without any errors.*

(ii) This question is modified from `Jones O. et al.`, 4.6.2. and reproduced here.

The function `order(x)` returns a permutation of `1:length(x)` giving the order of the elements of `x`. For example:

```
> x = c(1.1,0.7,0.8,1.4)
> y = order(x)
 [1] 2 3 1 4
> x[y]
[1] 0.7 0.8 1.1 1.4
```

Using `order` or otherwise, write a function that takes two inputs - a table and a number, such that the output is a file ordered (in increasing order) by the column represented by the number. Concretely, use the following layout:

```
order_fn<-function( table_name, num){
  ## Write your code here
}
```

If we had following data stored in a table called `this_table`:

```
ID      Age      Num Teeth
1       18       28
2       19       27
3       17       32
4       16       31
```

then we should get the following outputs:

```
> order_fn(this_table,1)
ID      Age      Num Teeth
1       18       28
2       19       27
3       17       32
4       16       31

> order_fn(this_table,2)
ID      Age      Num Teeth
4       16       31
3       17       32
1       18       28
2       19       27

> order_fn(this_table,3)
ID      Age      Num Teeth
2       19       27
```

```
1       18      28
4       16      31
3       17      32
```

Save your code in a file `order_fn.r`, and upload this to CMS. Keep the function name and the inputs the same as above in order for the grading script to work. We may test this script on a dataset with more than 3 columns, so do make sure that your function can work with all cases.

You can and should test your function using the data from `age_teeth.txt`.

In most practical situations involving data analysis, you may sometimes want to order the values in a column, and might not be able to use other database tools like Excel. Thus, having a function `order_fn` would be extremely helpful, since it will re-order the columns for you directly. In future work (whether in this course or otherwise), feel free to use the function you have just created.

Optional part

However, you might want to go further. Here is an optional exercise which will not be graded. Suppose you want to order data by several columns. For example, you might want to order data by the second column, then by the third column. Modify `order_fn` such that it takes in two inputs, the data in a table, and a vector consisting of the columns you want to order by. This vector could be of any length. For example, if we had:

```
ID      Age     Num Teeth
1       18      28
2       18      27
3       17      32
4       17      31
```

then

```
> order_fn(this_table, c(2,3))
```

```
ID      Age     Num Teeth
4       17      31
3       17      32
2       18      27
1       18      28
```

If you intend to do further data analysis using `R`, you may at times want to order tables by a few columns - this function would be extremely useful!

(iii) *Loops and the Harmonic Series*

This question assesses your proficiency in creating *for* and *while* loops, together with writing simple functions with regards to the Harmonic Series. Recall that the Harmonic Series is given by:

$$H_n = \sum_{k=1}^{n} \frac{1}{k}$$

so for example:

$$H_4 = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

and in general:

$$H_n = 1 + \frac{1}{2} + \ldots + \frac{1}{n}$$

This question mostly requires code together with a small writeup. Thus, put all your code and the write-up into an `.r` file named `harmonic.r`. Do ensure that it is easy for the grader to see which code / writeup corresponds to which part.

(a) Suppose you want to find H_n given a certain n. Write a `for` loop to find this, within the following function:

```
FindHn<-function(n){

# Put your loop here

}
```

Put your function in the file `harmonic.r`.

(b) Adam wants to find H_{100}, but instead of using the function you wrote in a), would prefer to use a while loop. He writes the following code:

```
counter = 1
mysum = 0
while(counter){

## Adam gets stuck here :(
## Fill in your code here (without changing the first few lines)
## to get the value of H_100

}
```

but gets stuck within the *while* loop environment. Without changing this structure, write code that will give the value of H_{100}. Put your code in the file `harmonic.r`.

(c) A more appropriate use of the *while* loop however, would be to check when a certain condition is fulfilled. For example, a while loop could be used to determine the smallest value of n where $H_n > B$, for some B. For example, if $B = 4$, then $n = 31$. Write a *while* loop to determine the smallest n for any B, within the following function:

```
FindN<-function(B){

# Put your loop here

}
```

Put your function in the file `harmonic.r`.

(d) Now suppose you write a never terminating snippet of code in `R` which keeps on evaluating H_n (as n increases), and to print its current value every hour. In pseudo code, it would be of the form:

```
hval = 0
n = 0
Repeat forever
  Increase n by 1
  Add 1/n to the value of hval
  Display value of hval at every hour
```

Furthermore, suppose you are immortal (and have nothing better to do), so you observe the value of H_n (`hval`) every hour. What can you say about the long run behaviour of the value of H_n and why? Put your explanation in `harmonic.r`.

(e) In the first four parts of this question, we have looked at loops to compute the following:

Loop 1: Given n, find the value of H_n.

Loop 2: Given some B, find the (smallest) value of n such that $H_n > B$

???

For each loop, is it possible to find H_n given n (equivalently find smallest n given B) without resorting to loops?

If yes, give the code. If no, explain why.

Put your answer (code / explanation for both loops) in `harmonic.r`.

Ensure that your file `harmonic.r` has answers to all five parts, and upload the file to CMS.

Part Two - 20 points

Exploratory data analysis by visualizing multivariate data

Most data in introductory statistics courses are either one dimensional or two dimensional, which makes them easy to visualize. For example, we could plot one dimensional data on the real line, and create scatterplots of two dimensional data. This makes it easy to see patterns in our data set.

However, if we are given data with a large number of dimensions, we would want to find techniques we can use which will help us visualize and understand the relationships between the variables. In previous statistics courses, scatterplots are commonly used to look at pairwise correlations between variables. However, scatterplots are not helpful at all when several variables may be interdependent on each other. Are there any techniques that could help with this?

In this question, we will be looking at two methods: Chernoff Faces and Principal Component Analysis (PCA). We will be using `R` packages and commands to conduct most of the visualization for this homework, so you do not need to worry about writing lots of code for these methods.

Fun Fact: In CS / ML terms, these two methods would be classified as unsupervised learning.

You can either submit a html or pdf file for this part of the homework, using either R Markdown, or LaTeX ing up your work.

Part of this exercise will indirectly test your ability to read `R` documentation and help files, as well as experimenting with the example commands given. The motivation is as follows. If you intend to use `R` in the future whether for research or industry work, some of the time you will be relying on packages written by other people to do part of your computations. Therefore, it is desirable to inculcate the habit of reading documentation carefully and playing about with `R` commands until you get the desired result.

The Dataset

We will be using the dataset `olympic.csv`. This data set gives the performance of 34 men at the 1988 Olympics decathalon. The dataset has 34 rows and 11 columns.

The first 10 columns correspond to the events of the decathlon: 100 meters (`m100`), long

jump (`ljump`), shotput (`shot`), high jump (`hjump`), 400 meters (`m400`), 110-meter hurdles (`m110h`), discus throw (`disc`), pole vault (`pvault`), javelin (`jav`) and 1500 meters (`m1500`).

The last column corresponds to the final score of the particular individual at the Olympics.

Load this dataset into `R`.

Chernoff Faces

Chernoff Faces was proposed in 1973 by Herman Chernoff, as a way of displaying multivariate data. By capitalizing on the fact that human beings are very good at recognizing faces, he let each feature of the human face (shape of nose, shape of eyes, hair length, etc) correspond to a variable in the dataset. As each variable in the dataset varies, the corresponding feature of the face varied as well.

This method isn't used widely these days, since most datasets have many more variables than potential features on a face[1], but since our Olympic dataset has only 11 columns, we'll try this out.

We will use the package `aplpack`. The manual can be found at `https://cran.r-project.org/web/packages/aplpack/aplpack.pdf`.

Thus, you need to install and load the package. After doing so, here are your following tasks.

1. By reading the manual, find a way to implement Chernoff Faces for the first ten columns of the dataset. In other words, implement Chernoff Faces for the ten Olympic events.

   Include the code you use and the image of the Chernoff Faces in your writeup.

2. As there are ten events, there should be ten features on the faces.

   Write down the events which the features correspond to, and describe how they vary in the writeup. Here is an example.

   The long jump corresponds to the length of the hair. The further the distance jumped, the longer the length of the hair.

3. Based on the Chernoff Faces you've generated, can you tell anything about the data? For example, if you see a group of faces (perhaps Face 3, 6, 10, 14) having similar features, what would that say about the corresponding variables?

   Write down your observations. Do they match with the data given?

[1] But more practically, humans can only absorb up to so many features.

4. There is one outlier in the data given, and you can spot it as one face generated will be extremely different compared to all the other faces.

   Which face is it, and why is it an outlier?

Principal Component Analysis

While Chernoff Faces are useful for determining the relationships between variables in a dataset at a glance, they're not very precise. We thus move on to Principal Component Analysis.

Some of you might be familiar with this, thus feel free to skip ahead to your task. If you've not heard of Principal Component Analysis, here's a brief explanation of what it is.

With multivariate data, we are interested in finding linear combinations of the variables in the data that give maximal variability. Why is this important? Let's consider a case in two dimensions.

Let's look at books from Amazon[2]. Some pairs of variables are interesting, and some are not so interesting. Suppose we looked at 100 books from Amazon, and considered the scatterplot of the variables `book price` versus `number of pages`. Consider rotating a line through the centroid of the scatterplot in 2D space, finding the "best line" that accounts for most of the variability.

Now suppose we look at the scatterplot of `number of pages` and `height`, and we again rotate a line through the centroid of the scatterplot in 2D space, and find the "best line" that accounts for most of the variability.

We'd expect that there's "more variability" in the scatterplot of `book price` versus `number of pages`.

We want to extend this idea to the multivariate case - and say something about the variation in the data based on some linear combinations of our variables. In this case, we'd be rotating a line not in 2 dimensional space, but n dimensional space.

[2]Those of you who have done STSCI / ILRST 2100 might remember the book data project.

Getting the Principal Components

Suppose X is the data matrix corresponding to the data in `olympic.csv`, up to the first 10 columns (we ignore the score).

First, we remove the outlier which we have detected in the previous part. olympic[-34,]

Secondly, we compute the correlation matrix of X.

Third, we compute the eigenvectors and eigenvalues of the correlation matrix using the `R` function `eigen`.

These eigenvectors[3] correspond to the principal components of X.

Write code to do the above, and put this in the R Markdown file. Display the output of `eigen` in the R Markdown file as well.

The output of `eigen` comes in two parts: the eigenvalue and its corresponding eigenvector. `R` orders the eigenvalues from the largest to the smallest, with $\lambda_1 > \lambda_2 > \ldots > \lambda_{10}$. With this ordering, then the i^{th} principal component is the i^{th} eigenvector corresponding to λ_i.

Interpreting the Principal Components

To check that you have the right output, λ_1 ought to be approximately 3.41, and the first eigenvector should be:

```
            [,1]
 [1,]  0.4158823
 [2,] -0.3940515
 [3,] -0.2691057
 [4,] -0.2122818
 [5,]  0.3558474
 [6,]  0.4334816
 [7,] -0.1757923
 [8,] -0.3840821
 [9,] -0.1799436
[10,]  0.1701426
```

[3]The proof of this can be found in most linear algebra / statistics / machine learning textbooks.

You can read each entries in this first eigenvector (first principal component) as the coefficients of the linear combinations of the columns of X which give maximal variance. This principal component accounts for $\frac{\lambda_1}{\sum_{i=1}^{10} \lambda_i}$ of the variance in the data.

Subsequently, we can think of the entries in the i^{th} eigenvector (i^{th} principal component) as the coefficients of the linear combinations of the columns of X which give maximal variance, *after accounting for the effect of the* $(i-1)^{th}$, $(i-2)^{th}, \ldots, 1^{st}$ *principal component.* This principal component accounts for $\frac{\lambda_i}{\sum_{i=1}^{10} \lambda_i}$ of the variance in the data.

Which Principal Components are important?

In general, we should look at the principal components accounting for most of the variance, and we can do so by looking at the *scree plot*. The scree plot is basically a plot of the (cumulative) variance accounted for by the principal components, and we will construct such a plot.

For example, the proportion of cumulative variance accounted for by the first principal component would be $\frac{\lambda_1}{\sum_{i=1}^{10} \lambda_i}$, and in general, the proportion of cumulative variance accounted for from the first k principal components would be $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{10} \lambda_i}$.

Thus, the x-axis of the scree plot would be the number of principal components, and in this case, it would range from 1 to 10.

The y axis on the other hand, would be the proportion of variance, and this ranges from 0 to 1.

With the scree plot, we usually look at the first k principal components which explain at least 80% of the variance[4].

Write code which creates a scree plot and draws a horizontal line at $y = 0.80$. Put both code and scree plot in the R Markdown file. Don't forget to label the plot.

We will look at the first four principal components.

[4]This isn't a rule set in stone - and this usually varies depending on the situation. There's also an elbow rule in determining which principal components to look at - basically find the cut-off where the shape of the curve looks like an elbow.

How do I interpret these principal components again?

Recall that the first principal component is given by:

```
            [,1]
 [1,]  0.4158823   # m100
 [2,] -0.3940515   # ljump
 [3,] -0.2691057   # shot
 [4,] -0.2122818   # hjump
 [5,]  0.3558474   # m400
 [6,]  0.4334816   # m110h
 [7,] -0.1757923   # disc
 [8,] -0.3840821   # pvault
 [9,] -0.1799436   # jav
[10,]  0.1701426   # m1500
```

If we look at the magnitude and the signs of each entry, we see that the events which focus on running have a positive sign (and large in magnitude except the `m1500`), whereas the other events have a negative sign.

This would roughly mean that high values (in the data) for events that focus on running are negatively correlated with low values for events that don't focus on running, and vice versa.

But this makes sense (remember that for running, a lower time taken means a better time) - good athletes would have low running times, but high scores in other events, and vice versa.

Caveat: Principal component analysis can also show what is *obvious* in the data.

We'll now take a look at the other three principal components. By looking at the signs and magnitudes of the entries in these principal components, give an interpretation of what they may represent. Does your interpretation correspond to the data you have (you can quickly eyeball the 34 entries to see if the interpretation makes sense). If you think there's no interpretation (eg, it's just noise), then state it. Put this in your writeup.

What else can I do with the principal components?

If we look at the first k principal components, we can think of them spanning a k dimensional subspace containing the "best" view of the data (in k dimensions). Thus, we can project our data X into this k dimensional subspace.

To do this in general, let P be a $10 \times k$ matrix, where each column of P corresponds to the corresponding principal component. Then, the matrix product XP represents the data in the k dimensional subspace.

We denote PC k to be the k^{th} principal component.

Let us consider **three** subspaces, each in $\mathbb{R}^2$. The subspace spanned by {PC1 and PC2}, {PC1 and PC3}, and {PC2 and PC3}.

Make a scatterplot of the points in each of these subspaces. Don't forget to label each plot (eg, PC1 against PC2) separately. Describe and give an interpretation of what you see in each plot.

A brief comparison

Did you find any similarities between your observations using Chernoff Faces and PCA? Give a brief explanation.