Statistics 506 Exam 1

October 21, 2015

For your information:

- `apply(x, margin, function)` – margin is "1" for rows and "2" for columns.

1. The "median absolute deviation" (MAD) of a set of numbers $x_1, \ldots, x_n$ is the median of the absolute values of the deviations of the numbers from their median. That is, it is

$$\text{med}|x_i - \text{med}\, x_j|.$$

R has functions called `median` and `abs` for calculating the median of all elements in an array, and the absolute value of each element in an array.

(a) If `x` is a one-dimensional array, write R code that will produce the MAD of the values in `x`.

Solution:

```
median(abs(x - median(x)))
```

(b) Suppose `x` is a two-dimensional array. Write R code that will produce a vector `v` such that `v[j]` contains the MAD of the j^{th} column of `x`.

Solution:

```
md = apply(x, 2, median)
z = t(t(x) - md)
z = abs(z)
v = apply(z, 2, median)
```

Note that if you wanted to check this against the built-in `mad` function in R, you would need to use the keyword argument `constant=1`, e.g.

```
v2 = apply(x, 2, function(x)mad(x, constant=1))
all.equal(v, v2)
```

(c) Supposing that you have constructed `v` as in part (b), write R code that will determine the number of columns of `x` whose MAD exceeds 5.

Solution:

```
sum(v > 5)
```

2. Suppose we are using R and have the following arrays:

$$\texttt{x} = \begin{pmatrix} 1 & 3 & 2 & 0 \\ 0 & 4 & 4 & 2 \end{pmatrix}$$

$$\texttt{v} = \begin{pmatrix} 2 & 5 \end{pmatrix}$$

$$\texttt{w} = \begin{pmatrix} 3 & 1 & 2 & 0 \end{pmatrix}$$

(a) What is `x - v`?

Solution:

$$\texttt{x} = \begin{pmatrix} -1 & 1 & 0 & -2 \\ -5 & -1 & -1 & -3 \end{pmatrix}$$

(b) What is `x - w`?

Solution:

$$\texttt{x} = \begin{pmatrix} -2 & 1 & -1 & -2 \\ -1 & 4 & 3 & 2 \end{pmatrix}$$

(c) What is `w - v`?

Solution:

$$1 \quad -4 \quad 0 \quad -5$$

3. Note that this question only involves Stata. Do not use any R in your response to this question. Suppose we have a text file containing data on single car accidents. Each record in the data set contains the following two variables:

- `speed`: the speed at which the vehicle was traveling at the moment of the accident, in miles per hour (MPH).
- `roadtype`: the road type on which the accident occurred; possible values are "freeway", "urban", "rural", and "local".

(a) A researcher wishes to create a new speed variable denoted `speed_ms`, containing the `speed` variable in units of meters per second. One mile per hour is equivalent to 0.44704 meters per second. Write Stata code so that `speed_ms` becomes available for use in the current Stata session.

Solution:

```
gen stata_ms = 0.44704*speed
```

(b) Describe the purpose of the following code. Be sure to describe the role of both lines of code.

```
encode roadtype, gen(rt)
mean speed_ms, over(rt)
```

Solution:

Line 1: Since `roadtype` was a string variable in the data file, it will most likely be a string variable when you load it into Stata. To stratify on a variable in Stata, it needs to be numeric. The `encode` statement creates a new numeric variable called `rt` that represents the four road types as integers.

Line 2: This statement tells us the mean speed for accidents that occured on each of the four road types. The speeds are presented in meters per second.

4. (a) Suppose that in R we have a one-dimensional array `x` = [3, 1, 2, 4, 6, 8]. Write code in two different ways that converts `x` into the 2×3 array

$$\begin{matrix} 3 & 2 & 6 \\ 1 & 4 & 8 \end{matrix}$$

Solution: One solution is to assign dimensions to the `dim` attribute of the array:

```
dim(x) = c(2, 3)
```

Another approach is to use the `array` function:

```
array(x, c(2, 3))
```

(b) Suppose we have a function in R defined using

```
f = function(x){2*x}
```

What is `f(f(f(x)))`, where `x` is the value that results from the code that you wrote in part (a)?

Solution:

$$\begin{matrix} 24 & 16 & 48 \\ 8 & 32 & 64 \end{matrix}$$

(c) What value of `z` results from running the following R program, where `x` is the value that results from the code that you wrote in part (a)?

```
f = function(x){function(y){y+x}}
g = f(3)
z = g(x)
```

Solution:

$$\begin{matrix} 6 & 5 & 9 \\ 4 & 7 & 11 \end{matrix}$$

5. (a) Briefly define what is meant for a data structure to be "homogeneous".

Solution: In a homogeneous data structure, all the values have the same type.

(b) Suppose the following pseudocode produces a value of "3". What type of assignment semantics does this language have?

```
x = [3, 5, 2]
y = x
y[1] = 99
print(sum(y == x))
```

Solution: This language has reference semantics, since changing an element of `y` changes the corresponding value in `x`. We know this since the two arrays must contain the same numbers, since the result of the final statement is "3". If this language had value semantics, like R, the final print statement would have produced "2".

6. Suppose we have a system with the following properties:

- Pointers (memory addresses used for indirection) and numbers occupy 8 bytes, and strings occupy 4 bytes per character.
- Homogeneous arrays of numbers are stored contiguously, while homogeneous arrays of strings, and all inhomogeneous arrays are stored using indirection.
- The total storage required for an array is equal to the storage for its contents, plus an overhead of 8 bytes (to store the length of the array).
- There is no overhead for storing scalars (i.e. we only need to count the memory needed to store the scalar value itself).

In this system, we wish to store the following array:

```
[34, ['cat', 'dog'], [56, 89]]
```

(a) Describe in words or use a sketch to illustrate how this data object might be laid out in memory. State how much total storage the data object would occupy in our system.

Solution: The top-level array is laid out as:

```
[pointer, pointer, pointer]
```

which requires 8 bytes of overhead + 24 bytes of pointer storage, for a total of 32 bytes.

The value pointed to by the first element of the top level array is a number which requires 8 bytes of storage.

The value pointed to by the second element of the top level array is an array of strings. Since all arrays of strings use indirection, this array will be laid out as

```
[pointer, pointer]
```

which requires 8 bytes of overhead + 16 bytes of pointer storage. In addition, the strings themselves require 24 bytes of storage, so the total storage for the ['cat', 'dog'] array is 48 bytes.

The vaoue pointed to by the third element of the top-level array is an array of two numbers, which does not require indirection. This array requires 8 bytes of overhead + 16 bytes of data storage, for a total of 24 bytes.

Thus, the overall data structure requires 32 + 8 + 48 + 24 = 112 bytes.

An alternative solution that is also acceptable is to add 8 bytes of overhead for the strings (viewing them as arrays of characters), which would add 16 bytes, for a new total of 128 bytes.

(b) Write R code to represent this data object in a way that is most consistent with the representation given above.

Solution:

```
list(1, c('cat', 'dog'), c(56, 89))
```

7. Suppose we have a dataset containing records for patients who were admitted into one of several medical facilities. For each patient, we have their "medical record number" (denoted `MRN`), which is a unique patient identifier, the name of the facility to which they were admitted, denoted `facility`, the name of the department to which they were admitted, denoted `department`, and a binary variable denoted `died` indicating whether the patient died while in the facility, coded 1 for a death and 0 otherwise.

(a) Our first goal is to produce a table showing the mortality rate (the fraction of patients who died) within each facility. The code below produces the correct result, but is inefficient and needlessly complex. Describe one concrete aspect of this code that is computationally inefficient.

```
library(dplyr)

df = read.csv('data.csv')

rslt = list()

for (f in df$facility) {
    gb = group_by(df, facility)
    su = summarise(gb, mortality=mean(died))
    ii = (su$facility == f)
    rslt[[f]] = su$mortality[ii]
}
```

Solution: There are at least two major ways in which this code is computationally inefficient:

- `df$facility` contains each facility name multiple times, so we are repeatedly calculating the same mean values.
- The `group_by` and `summary` statements do not depend on the loop variable `f`, therefore these could be moved out of the loop with no effect on the results.

(b) Continuing with part (a), rewrite the code to be as simple and efficient as is prac-

tically possible.

Solution: The following code will give the desired results:

```
gb = group_by(df, facility)
su = summarise(gb, mortality=mean(died))
```

(c) Describe in words the result of running the following code:

```
library(magrittr)

df %>% filter(died==1) %>%
    group_by(facility, department) %>%
    summarize(n=n()) %>%
    mutate(f=n/sum(n))
```

Solution: This code gives us the mortality rate within each department of each facility.

(d) Continuing with part (c), write equivalent code that does not use pipes.

Solution:

```
df1 = filter(df, died==1)
df2 = group_by(df1, facility, department)
df3 = summarize(df2, n=n())
df4 = mutate(df3, f=n/sum(n))
```

8. Create a simple artificial example of a dataset with no more than 6 records in long format, then express the same dataset in wide format. Note: you are not writing any code here, you are making up the data and variable names to illustrate the difference between long and wide format data.

Solution: Here is the dataset in long format:

Subject	Hand	Fingers
1	Left	5
1	Right	5
2	Left	4
2	Right	5
3	Left	5
3	Right	0

Here is the same dataset in wide format:

Subject	Left	Right
1	5	5
2	4	5
3	5	0