Statistics 506 Exam 2

December 17, 2015

1. (a) Suppose that `li` is a list containing K arrays, each of which consists of distinct integers that lie between 1 and n. That is, for each $k = 1, \ldots, K$, `li[[k]]` is an array containing distinct values in the set $\{1, \ldots, n\}$. Describe the meaning of the values contained in `f` and `g` after running the following R program.

```
f = array(0, n)
for (k in 1:K) {
    f[li[[k]]] = f[li[[k]]] + 1
}
g = sum(f == 0)
```

Solution: `f` is an array such that `f[i]` is the number of times that the value `i` appears in the arrays in `li`. `g` is a boolean variable that is `true` if all the integers in $\{1, \ldots, n\}$ are contained at least once within the elements of `li`.

(b) Suppose that `tmax` is a 1000×31 array such that `tmax[i, j]` contains the maximum temperature recorded at station `i` on the `j`th day of January in 2014, in degrees Celsius. Describe the result obtained by running the following R program.

```
tmax1 = tmax[, 2:31] - tmax[, 1:30]
tmax2 = rowMeans(tmax1 > 0)
print(sum(tmax2 > 0.75))
```

Solution: The result is the proportion of stations such that on at least 75% of days in January, the maximum temperature was higher than on that day compared to the preceeding day.

(c) What is the result of running the following program?

```
x = 3
f = function(y){y+x}
g = function(y){x=10; f(y)}
g(7)
```

Solution: The result is 10.

2. Below you are asked to write short R programs to accomplish specific tasks. It is fine if your code is not exactly syntactically correct R code. Focus on the logic of the algorithm, and restrict your code to using basic R syntax involving array processing.

(a) Suppose we have a 1-dimensional array `v` that contains only 0's and 1's. Our goal is to efficiently find the length of the longest subsequence of consecutive 1's in `v`. Write vectorized R code to accomplish this goal that does not use any loops or use the `apply` function.

Solution:

```
v = c(0, v, 0)
n = length(v)
i1 = which((v[1:(n-1)] == 0) & (v[2:n] == 1))
i2 = which((v[1:(n-1)] == 1) & (v[2:n] == 0))
len = i2 - i1
maxlen = max(len)
```

(b) Suppose we have a $n \times 2$ array `z` such that `z[i, 1]` contains the year that a person was born, and `z[i, 2]` contains the year that a person died. All the years are whole numbers between 1 and 2015. Write R code to determine the number of years between year 1 and year 2015 during which nobody in the dataset was alive.

Solution:

```
num_alive = array(0, 2015)
for (i in 1:dim(z)[1]) {
    num_alive[z[i,1]:z[i,2]] = num_alive[z[i,1]:z[i
        ,2]] + 1
}
print(which(num_alive == 0))
```

(c) Suppose we have a $n \times 2$ array `Z` of points in the plane (e.g. `Z[i,]` are (x, y) or (latitude, longitude) coordinate pairs). We have another array `B` that is $m \times 4$ and contains the locations of boxes in the plane. For each row index `i`, the tuple (`B[i, 1]`, `B[i, 2]`) contains the coordinates of the lower left corner of the box, `B[i, 3]` is the width of the box, and `B[i, 4]` is the height of the box. Write an R program that identifies for each point in `Z` the index of the first box in `B` that contains the point. The index value should be `NA` when no box contains the point. Do not use any nested `for` loops.

Solution:

```
Q = array(NA, dim(Z)[1])
```

```
for (i in 1:dim(Z)[1]) {
    ii = (Z[i, 1] >= B[i, 1])
    ii = ii & (Z[i, 1] <= B[i, 1] + B[i, 3])
    ii = ii & (Z[i, 2] >= B[i, 2])
    ii = ii & (Z[i, 2] <= B[i, 2] + B[i, 4])
    ii = which(ii)
    if (length(ii) > 1) {
        Q[i] = ii[1]
    }
}
```

3. Below are three examples of SAS or R code that use pipes. Describe in words the steps that the computer is following when executing these lines of code. Focus on what is common among the three examples, specifically with respect to the use of the pipe (i.e. don't focus on what 'fread', 'read.csv', or 'infile' are doing). Write something that is *specific*, and if you are not certain about exactly what is happening, write something that is *plausible*. Use appropriate terminology learned in this course.

```
# Example 1 (R)
df = fread('gzip -dc 2014.csv.gz')

# Example 2 (R)
df = read.csv(pipe('gzip -dc 2014.csv.gz'))

// Example 3 (SAS)
filename fid pipe 'gzip -dc data.csv.gz' lrecl=80;

data ds;
    infile fid delimiter=',';
    input name $ height weight;
```

Solution: 'gzip' is a program that runs as an independent process, separate from the R or SAS process. To create the pipe, the operating system sets up a memory buffer that is shared between the two processes. 'gzip' reads the file as a stream, decompresses the beginning of it, and places the first part of the decompressed data into the memory buffer. The other end of the pipe (SAS or R) is able ro read from the memory buffer. As the data are read from the buffer, new data produced by gzip take its place until the entire file has been read.

4. Suppose we have a collection of 200 newspaper articles from 2014, and a similar collection from 1974. Provide brief answers to the following questions.

(a) Define a "document term matrix".

Solution: Suppose we have a collection of documents. A DTM is a matrix whose rows correspond to the documents and whose columns correspond to terms (words) that appear in the documents (it can also be the transpose of this matrix). The i, j entry of the DTM is the number of times that term j appears in document i.

(b) What does it mean for two columns of a document term matrix to be highly correlated?

Solution: The two terms corresponding to the two columns appear in similar sets of documents.

(c) What does it mean for two rows of a document term matrix to be highly correlated?

Solution: The two documents corresponding to the two rows contain similar terms.

(d) What does it mean for two columns of a document term matrix to have similar average values?

Solution: The two terms occur with similar frequencies in the document collection.

(e) What does it mean for two rows of a document term matrix to have similar average values?

Solution: The two documents have similar length.

(f) Devise a simple but meaningful way to assess the similarity between the two collections of articles. Write a sketch of R code to perform the calculations needed for your assessment.

Solution: Let `dtm1` and `dtm2` denote the document term matrices for the two collections of documents. Suppose they have been ordered so that the columns correspond to the same terms in the same order. We could take the mean term frequency within each collection of documents, then consider the average squared difference between the two means:

```
terms1 = slam::col_sums(dtm1)
terms2 = slam::col_sums(dtm2)
x = mean((terms1 - terms2)^2)
```

5. Suppose you run the following program, using the 2014 GHCN data set that we used in class. Below the code are four possibilities for the first 3 rows of `df5`. Which of these can be the real result of the program? You do not need to justify your answer.

```
df = read.csv(pipe('gzip -dc 2014.csv.gz'))
names(df) = c('station', 'date', 'vtype', 'value', 'w', 'x',
    'y', 'z')
df1 = filter(df, vtype=='TMAX') %>% mutate(value=value/10)
df2 = filter(df, vtype=='TMIN') %>% mutate(value=value/10)
df3 = group_by(df1, station) %>% summarize(v1=max(value) -
   min(value))
df4 = group_by(df2, station) %>% summarize(v2=max(value) -
   min(value))
df5 = inner_join(df3, df4, by='station')
```

Solution: The correct answer is c.

a.

```
      station    v1     v2
       (fctr) (dbl)  (dbl)
1 AE000041196  28.5  -23.0
2 AEM00041194  26.1  -21.3
3 AEM00041217  28.8 - 23.4
```

b.

```
      station   vbl   val
       (fctr) (dbl) (dbl)
1 AE000041196   v1   28.5
2 AE000041196   v2   23.0
3 AEM00041194   v1   26.1
```

c.

```
      station    v1    v2
       (fctr) (dbl) (dbl)
1 AE000041196  28.5  23.0
2 AEM00041194  26.1  21.3
```

```
3 AEM00041217  28.8  23.4
```

d.

```
      station    v1    v2
       (fctr) (dbl) (dbl)
1 AE000041196   285   230
2 AEM00041194   261   213
3 AEM00041217   288   234
```

e.

```
      date    v1    v2
    (fctr) (dbl) (dbl)
1 20140101  28.5  23.0
2 20140102  26.1  21.3
3 20140103  28.8  23.4
```

6. Spark programs process data structures called "Resilient Distributed Datasets" (RDDs). State and define three specific properties of RDDs that allow them to be an efficient mechanism for processing large datasets. For each property, state and define the property, then briefly say why this property is advantageous.

Solution: Some possible properties are:

They are *in-memory*, meaning that if possible all the data in an RDD is stored in the computer's volatile (RAM) memory, not in stable (e.g. disk-based) memory.

They are *distributed*, meaning that they are divided into partitions that utilize the memory of multiple computers. This allows them to be much larger than the RAM of any one computer.

They are *immutable*, meaning that once created they cannot be changed. This allows them to be distributed without needing to worry about updating data on one machine due to changes that are made on a different machine.

They are *resilient*, meaning that the information needed to create an RDD is stored redundantly on multiple machines. If one machine goes down, the RDD can be reconstructed from data on other nodes.

They are *lazily constructed*, meaning that the operations needed to create an RDD are not executed until the RDD is required by an action. This allows them to be communicated efficiently over

7. Suppose we have a dataset of medical billing records containing five variables. The first few lines of the data file are as follows (the actual file is comma-separated file named 'data.csv', but it is shown below as a fixed position file for clarity).

```
provider_id     patient_id     procedure_code     age     sex    outpatient
31              4213           343                35      f      1
31              4213           299                35      f      1
32              4213           512                35      f      1
34              4217           342                41      m      1
```

Below is a SAS program that operates on this dataset. Note that 'count' is an aggregation function that give the number of non-missing values in each group for one variable.

```
proc import datafile='data.csv' out=data1;

proc sql;

    create table table1 as
        select age, procedure_code, sex, count(age) as n from data1
        where outpatient = 1
        group by age, procedure_code, sex;

proc transpose data=table1 out=table2;
    by age procedure_code;
    id sex;
    var n;

data table3;
    set table2;
    if f eq . then f = 0;
    if m eq . then m = 0;
    if f + m > 1000;
    fp = f / (f + m);
    x = 2*fp - 1;

proc sort data=table3 out=table4;
    by x;

proc print data=table4(obs=5);

run;
```

(a) Give a numerical example that shows what the output might look like. Use made-up numbers, not the data shown above.

Solution:

```
obs    age    procedure_code    f        m       fp      x
1      30     13124             99000    1000    0.99    0.98
2      29     13124             98000    2000    0.98    0.96
3      57     14132             97000    3000    0.97    0.94
4      52     14132             96000    4000    0.96    0.92
5      25     13125             95000    5000    0.95    0.90
```

(b) Briefly describe how the results should be interpreted, and the research goal for which these results would be relevant. Be sure to explain (briefly) every important aspect of the analysis implemented in the above code.

Solution: The focus here is on gender differences in the utilization of medical procedures. The analysis is stratified by age. The variable called `x` is the difference between the proportion of females and the proportion of males who recieve a given type of procedure. The output will be the five procedure/age pairs for which this difference is greatest, meaning that females recieve the procedure much more often than males, within a given age range.

8. Suppose we have a dataset of bigrams, which are pairs of consecutive words that appear in a text document. The words have been tagged for their part of speech, e.g. 'dog' is a noun. A small excerpt of the data is:

```
hot    dog    adjective    noun
dog    food   noun         noun
run    fast   verb         adverb
to     be     preposition  verb
```

Suppose these data have been loaded into a Spark RDD called `data1`. Describe what each of the following program fragments computes (remember that Python arrays are indexed starting from zero, and the length of a string variable in Python is the number of characters in the string).

1.

```
data2 = data1.map(lambda x : [x[0], 1])
data3 = data2.reduceByKey(lambda x,y : x+y).collect()
```

Solution: This computes the number of times that each word appears.

2.

```
data2 = data1.map(lambda x : [(x[2], x[3]), 1])
data3 = data2.reduceByKey(lambda x,y : x+y).collect()
```

Solution: This computes the number of times that each pair of parts of speech occur consecutively.

3.

```
data2 = data1.map(lambda x : [x[2], len(x[0])])
data3 = data2.reduceByKey(lambda x,y : max(x, y)).collect()
```

Solution: This computes the maximum length of all words that belong to each part of speech.

4.

```
data2 = data1.map(lambda x : [x[2], [len(x[0]), 1]])
data3 = data2.reduceByKey(lambda x,y : x[0]+y[0], x[1]+y[1])
#data4 = data3.map(lambda x : (x[0], x[1] / x[2])).collect()
data4 = data3.map(lambda x : (x[0], x[1][0] / x[1][1])).collect()
```

Solution: This was supposed to compute the mean length of all words with a given part of speech. Sorry, but note the mistaken line (commented out with '#') and the correct line below it.

9. In R, a `data.table` object can be indexed, but an R `data.frame` object is not indexed. This means that data values in a `data.frame` object can be located using a binary search – if we are searching for a row whose index variable has value y, we first check the middle row, whose index variable is y_0, and if $y_0 < y$ we restrict the search to the rows after the middle row, whereas if $y_0 > y$ we restrict our search to the rows before the middle row. This search strategy is repeated until the row of interest is located.

   Starting from an indexed data structure of a given size n, how big does the data structure size n' need to become in order for searches to double in time?

Solution: The size of the data remaining to search is cut in half during each iteration of the process described above. Therefore it takes $\log_2(n)$ steps to find the target value. If $n' = n^2$, then the search time doubles.