THE UNIVERSITY OF AUCKLAND

SECOND SEMESTER, 2016
Campus: City

STATISTICS

Data Science Practice

(Time allowed: THREE hours)

NOTE:
- This examination consists of **14** questions.
- The marks for all questions sum to **100**.
- Questions are **NOT** worth equal marks.
- You should attempt **ALL** questions.
- For questions where you are required to write computer code, if you do not know the exact code, you can still gain some of the marks by writing an approximation of what the code should be.

1. **[5 marks]**

Figure 1 shows the content of a JSON file, `"AT.json"`, and the following code reads this file into R and shows the resulting R object, `trips`.

```
> library(jsonlite)
> trips <- fromJSON("AT.json")
> trips

    id stop_time_update.stop_sequence stop_time_update.stop_id  timestamp
1 2928                             20                     7812 1474316682
2 2929                             60                     6569 1474316645

> dim(trips)

[1] 2 3

> names(trips)

[1] "vehicle"          "stop_time_update" "timestamp"
```

Explain what sort of R object has been created and **write R code** to extract the `stop_id` information from the R object `trips`.

```
[
    {
        "vehicle": {
            "id": "2928"
        },
        "stop_time_update": {
            "stop_sequence": 20,
            "stop_id": "7812"
        },
        "timestamp": 1474316682
    },
    {
        "vehicle": {
            "id": "2929"
        },
        "stop_time_update": {
            "stop_sequence": 60,
            "stop_id": "6569"
        },
        "timestamp": 1474316645
    }
]
```

Figure 1: The JSON file `"AT.json"`.

2. **[5 marks]**

Figure 2 shows the content of an XML file, `"IRD.xml"`.

Write down the result of the following R code:

```
> library(xml2)
> ird <- read_xml("IRD.xml")
> xml_text(xml_find_all(ird, "//td[@align = 'right']"))
```

Write R code to extract the first column of values from the table. Your code should produce the following result:

```
[1] "18 NCO Club"                    "1977 Masters Association"
[3] "1979 Reunion"                   "1993 Summer Camp Account"
[5] "1St Wainuiomata Venterer Unit" "44 South Travel"
[7] "81 Masters Association"
```

```
<table width="100%" cellpadding="0" cellspacing="0"><tbody>
  <tr>
    <td>18 NCO Club</td>
    <td align="right">$142.03</td>
  </tr>
  <tr>
    <td>1977 Masters Association</td>
    <td align="right">$359.77</td>
  </tr>
  <tr>
    <td>1979 Reunion</td>
    <td align="right">$532.77</td>
  </tr>
  <tr>
    <td>1993 Summer Camp Account</td>
    <td align="right">$1,308.78</td>
  </tr>
  <tr>
    <td>1St Wainuiomata Venterer Unit</td>
    <td align="right">$431.14</td>
  </tr>
  <tr>
    <td>44 South Travel</td>
    <td align="right">$489.60</td>
  </tr>
  <tr>
    <td>81 Masters Association</td>
    <td align="right">$221.08</td>
  </tr>
</tbody></table>
```

Figure 2: The XML file `"IRD.xml"`.

3. **[5 marks]**

Explain what each of the following shell commands is doing and, where there is output, what the output means:

```
pmur002@sc-stat-346130:/home/paul$ ssh stats769prd01.its.auckland.ac.nz

pmur002@stats769prd01:~$ mkdir exam

pmur002@stats769prd01:~$ cd exam

pmur002@stats769prd01:~/exam$ cp /course/data/Ass2/exreg-10000.* .

pmur002@stats769prd01:~/exam$ ls -lh
total 16M
-rw-rw---- 1 pmur002 pmur002 7.8M Sep 20 12:12 exreg-10000.bin
-rw-rw---- 1 pmur002 pmur002  473 Sep 20 12:12 exreg-10000.desc
-rw-rw---- 1 pmur002 pmur002 7.7M Sep 20 12:12 exreg-10000.txt

pmur002@stats769prd01:~/exam$ wc exreg-10000.txt
  10000 1010000 7979235 exreg-10000.txt

pmur002@stats769prd01:~/exam$ awk 'NR < 1000' exreg-10000.txt > exreg-sub.txt
```

4. **[5 marks]**

Explain the memory usage results from the following R code and output.

What is the significance of the `NULL` at the end of the function definition?

```
> gc(reset=TRUE)

         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 326457 17.5     592000 31.7   326457 17.5
Vcells 535537  4.1    1023718  7.9   535537  4.1

> x <- rnorm(1000000)
> gc()

          used (Mb) gc trigger (Mb) max used (Mb)
Ncells  326423 17.5     592000 31.7   328238 17.6
Vcells 1535490 11.8    2613614 20.0  1537444 11.8

> f <- function() {
+     x <- rnorm(1000000)
+     NULL
+ }
> f()

NULL

> gc()

          used (Mb) gc trigger (Mb) max used (Mb)
Ncells  326438 17.5     592000 31.7   330982 17.7
Vcells 1535523 11.8    2613614 20.0  2541622 19.4
```

5. **[5 marks]**

Figure 3 shows the first few lines of a CSV file called `"1987.csv"`. This file contains flight data from 1987 for flights within the USA.

```
Year,Month,DayofMonth,DayOfWeek,DepTime,CRSDepTime,ArrTime,CRSArrTime,UniqueCarr
ier,FlightNum,TailNum,ActualElapsedTime,CRSElapsedTime,AirTime,ArrDelay,DepDelay
,Origin,Dest,Distance,TaxiIn,TaxiOut,Cancelled,CancellationCode,Diverted,Carrier
Delay,WeatherDelay,NASDelay,SecurityDelay,LateAircraftDelay
1987,10,14,3,741,730,912,849,PS,1451,NA,91,79,NA,23,11,SAN,SFO,447,NA,NA,0,NA,0,
NA,NA,NA,NA,NA
1987,10,15,4,729,730,903,849,PS,1451,NA,94,79,NA,14,-1,SAN,SFO,447,NA,NA,0,NA,0,
NA,NA,NA,NA,NA
...
```

Figure 3: The first few lines of the CSV file `"1987.csv"`.

The following R code and output shows the time and memory requirements involved in naively reading the file `"1987.csv"` into an R data frame and then calculating the average flight departure delay for each day of the week.

```
> system.time(f1987 <- read.csv("1987.csv"))
   user  system elapsed
  8.785   0.176   8.967

> object.size(f1987)
152204024 bytes

> system.time(delays <- aggregate(f1987["DepDelay"],
                                  list(DoW=f1987$DayOfWeek),
                                  mean, na.rm=TRUE))
   user  system elapsed
  0.997   0.004   1.001

> delays
  DoW DepDelay
1   1 7.827491
2   2 9.086585
3   3 9.364805
4   4 8.143775
5   5 7.411825
6   6 6.034632
7   7 8.408912
```

Write R code that calls the `read.csv()` function, but with additional argument that would make the call run faster and create a smaller data frame than the code above. **Explain** why your code would be faster and use less memory.

Write R code that uses functions from the **data.table** package to read the file `"1987.csv"` into R much faster **and** to calculate the average departure delay for each day of the week much faster.

6. **[5 marks]**

The following R code and output shows the number of cores and the memory capacity of one of the virtual machines used in this course.

```
> detectCores()
[1] 20

> system("free")
             total       used       free     shared    buffers     cached
Mem:     206350080   42760988  163589092         44     262884   40465868
-/+ buffers/cache:    2032236  204317844
Swap:      1949692     131744    1817948
```

The following R code and output shows information about the full set of CSV files ("1987.csv" to "2008.csv") that contain US flight data over 22 years.

```
> system("ls -lh /course/data/ASADataExpo/*.csv")
-rw-r--r-- 1 pmur002 pmur002 122M Aug  4 14:48 /course/data/ASADataExpo/1987.csv
-rw-r--r-- 1 pmur002 pmur002 478M Aug  4 14:48 /course/data/ASADataExpo/1988.csv
-rw-r--r-- 1 pmur002 pmur002 464M Aug  4 14:48 /course/data/ASADataExpo/1989.csv
-rw-r--r-- 1 pmur002 pmur002 486M Aug  4 14:50 /course/data/ASADataExpo/1990.csv
-rw-r--r-- 1 pmur002 pmur002 469M Aug  4 14:47 /course/data/ASADataExpo/1991.csv
-rw-r--r-- 1 pmur002 pmur002 470M Aug  4 14:49 /course/data/ASADataExpo/1992.csv
-rw-r--r-- 1 pmur002 pmur002 469M Aug  4 14:49 /course/data/ASADataExpo/1993.csv
-rw-r--r-- 1 pmur002 pmur002 479M Aug  4 14:48 /course/data/ASADataExpo/1994.csv
-rw-r--r-- 1 pmur002 pmur002 507M Aug  4 14:48 /course/data/ASADataExpo/1995.csv
-rw-r--r-- 1 pmur002 pmur002 510M Aug  4 14:52 /course/data/ASADataExpo/1996.csv
-rw-r--r-- 1 pmur002 pmur002 516M Aug  4 14:49 /course/data/ASADataExpo/1997.csv
-rw-r--r-- 1 pmur002 pmur002 514M Aug  4 14:47 /course/data/ASADataExpo/1998.csv
-rw-r--r-- 1 pmur002 pmur002 528M Aug  4 14:48 /course/data/ASADataExpo/1999.csv
-rw-r--r-- 1 pmur002 pmur002 544M Aug  4 14:47 /course/data/ASADataExpo/2000.csv
-rw-r--r-- 1 pmur002 pmur002 573M Aug  4 14:48 /course/data/ASADataExpo/2001.csv
-rw-r--r-- 1 pmur002 pmur002 506M Aug  4 14:47 /course/data/ASADataExpo/2002.csv
-rw-r--r-- 1 pmur002 pmur002 598M Aug  4 14:50 /course/data/ASADataExpo/2003.csv
-rw-r--r-- 1 pmur002 pmur002 639M Aug  4 14:47 /course/data/ASADataExpo/2004.csv
-rw-r--r-- 1 pmur002 pmur002 640M Aug  4 14:49 /course/data/ASADataExpo/2005.csv
-rw-r--r-- 1 pmur002 pmur002 641M Aug  4 14:49 /course/data/ASADataExpo/2006.csv
-rw-r--r-- 1 pmur002 pmur002 671M Aug  4 14:50 /course/data/ASADataExpo/2007.csv
-rw-r--r-- 1 pmur002 pmur002 658M Aug  4 14:47 /course/data/ASADataExpo/2008.csv
```

The following R code could be used to read all 22 CSV files into R as a single data frame and calculate the average departure delay for each day of the week.

```
filenames <- paste0(1987:2008, ".csv")
flights <- do.call(rbind, lapply(filenames, read.csv))
aggregate(flights["DepDelay"],
          list(DoW=flights$DayOfWeek),
          mean, na.rm=TRUE)
```

Discuss the time and memory requirements that would be involved in running this R code and whether it would be able to run on the virtual machine.

7. **[10 marks]**

The function `sumFile()` (code not shown) takes one argument, the name of a CSV file, and calculates the sum of the departure delay column and counts the number of non-`NA` values in the departure delay column, for each day of the week. The result of calling this function on the file `"1987.csv"` is shown below.

```
> sumFile("1987.csv")
   DoW     sum  count
1:   1 1457291 186176
2:   2 1692113 186221
3:   3 1757905 187714
4:   4 1619129 198818
5:   5 1356023 182954
6:   6 1037999 172007
7:   7 1498897 178251
```

Write R code that calls the `sumFile()` function in parallel across multiple cores to calculate sums and counts for all CSV files in the data set **and** then combines the results from all CSV files to calculate the average departure delay for each day of the week across all 22 files. The final result would look like the output below.

```
> meanDepDelay
     [,1]      [,2]
[1,]    1  7.850057
[2,]    2  6.855870
[3,]    3  7.651197
[4,]    4  9.246910
[5,]    5 10.151539
[6,]    6  6.887023
[7,]    7  8.409293
```

Discuss the best way to schedule the 22 calls to `sumFile()` across the multiple cores (hint: think about whether each call to `sumFile()`, which handles a different CSV file, will take the same amount of time to run).

8. **[5 marks]**

The following code was used to profile the `sumFile()` function.

```
> Rprof("sumFile.log")
> sumFile("1987.csv")
> Rprof(NULL)
```

The profile results are summarised below using `summaryRprof()` ...

```
> summaryRprof("sumFile.log")
$by.self
          self.time self.pct total.time total.pct
"fread"        1.38    97.18       1.38     97.18
"!"            0.02     1.41       0.02      1.41
"forderv"      0.02     1.41       0.02      1.41

$by.total
               total.time total.pct self.time self.pct
"sumFile"            1.42    100.00      0.00     0.00
"fread"              1.38     97.18      1.38    97.18
"["                  0.04      2.82      0.00     0.00
"[.data.table"       0.04      2.82      0.00     0.00
"!"                  0.02      1.41      0.02     1.41
"forderv"            0.02      1.41      0.02     1.41

$sample.interval
[1] 0.02

$sampling.time
[1] 1.42
```

... and using `profReport()`.

```
> profReport("sumFile.log")
sumFile > fread
---------------
1.38

sumFile > [ > [.data.table > !
------------------------------
0.02

sumFile > [ > [.data.table > forderv
------------------------------------
0.02
```

Explain the profile results and what they tell us about how the `sumFile()` function works and where it spent most of its time.

9. **[5 marks]**

i. **Explain** why we might need to use the functions `GET()` or `POST()` from the **httr** package to access a web site, rather than just the `download.file()` function.

ii. **Explain** why we might need to use the functions `makeCluster()` and `clusterApply()`, instead of the `mclapply()` function.

10. **[10 marks]**

Table 1 shows the SS_{residual} and adjusted R^2 values for a model selection procedure of a linear regression problem with 6 input variables, $x_1, \ldots, x_6$. At step i, the model contains the input variable which is shown in column "Variables entered" and the variables at all previous steps. At step i, the SS_{residual} column shows the value of residual sum of squares after the variable in the "Variables entered" column has been added to the previous model (from step $i-1$). This value is the smallest SS_{residual} among all possible models that add a single predictor to the previous model (from step $i-1$).

1. Which of the model selection procedures is used in this problem?
2. Based on Table 1, what is the best linear predictor for this problem?

Step	Variables entered	SS_{residual}	Adjusted R^2
0	Intercept	34.026	0.700
1	x4	33.789	0.707
2	x3	33.584	0.703
3	x2	33.583	0.713
4	x5	33.586	0.714
5	x1	33.590	0.713
6	x6	33.595	0.701

Table 1: Summary of the model selection procedure

11. **[10 marks]**

The following code has been used to fit a k-nearest neighbor classifier for given data.

```
> library(class)
> train <- rbind(iris3[1:25,,1], iris3[1:25,,2], iris3[1:25,,3])
> test <- rbind(iris3[26:50,,1], iris3[26:50,,2], iris3[26:50,,3])
> cl <- factor(c(rep("s",25), rep("c",25), rep("v",25)))
> knnout <- knn(train, test, cl, k = 3)
> knnout

 [1] s s s s s s s s s s s s s s s s s s s s s s s s s c c v
[29] c c c c c v c c c c c c c c c c c c c c c c c v c c v v v
[57] v v v v v v v c v v v v v v v v v v v
Levels: c s v

> table(knnout,cl)

      cl
knnout  c  s  v
     c 23  0  3
     s  0 25  0
     v  2  0 22
```

Based on the output of the code, calculate

1. the misclassification rate for test data.
2. the sensitivity and specificity for test data.

12. **[10 marks]**

Kernel density estimation is an unsupervised learning procedure that estimates the probability density of a new observation x_0 by counting observations close to it with weights that decrease with distance from it. Formally, for a given univariate random sample $x_1,\ldots,x_n$ drawn from a probability density $g_X(x)$, kernel density estimation uses the following formula to estimate the probability density $g_X(x)$ of a new observation x_0,

$$\hat{g}_X(x_0) = \frac{1}{nh}\sum_{i=1}^{n} K_h(x_0, x_i)$$

where n is the sample size and h is a tuning parameter for the kernel function K.
Assume that a given learning sample $\mathcal{L}$, with one input variable, contains 100 data points from population I and 100 data points from population II. We want to classify the population of a new observation based on this learning set. This problem is a classification problem with one input variable and a response with two categories. Explain a classifier algorithm which uses kernel density estimation to classify the new observation.

13. **[10 marks]**

Explain the k-nearest neighbor algorithm for fitting models (a) when doing prediction of a continuous response, and (b) when doing classification.

14. **[10 marks]**

Explain the steps of how you would build a model for a classification problem when you receive a data set of 30,000 observations, 100 inputs and a categorical response variable.
