1.

(a)-F, will seperate each line into multiple variales according to ,

(b)$1 means the firlst variable after separation by comma , $2 means the second variable, $5 means the fifth varialbe, and "," will put a comma between two variables.

(c)

```
04-JUL-2016,01N00190,0
09-JUL-2016,08500161,0.5
08-SEP-2016,09400240,0
```

(e)

```
awk -F, -e '$3 == "H" { print $1","$2","$5 }' test-data.csv | wc -l
```

(f)

```
awk -F, -e '$3 == "H" { print $1","$2","$5 }' test-data.csv | sort
-t, -k 3,3 -nr | head - 6
```

2.

(a)

```
library(jsonlite)
testjson <- fromJSON('test-data.json')
testjson <- testjson$features$attributes
```

(b) //day and site/@id are Xpath expressions.

//day will get all day elements in xml file

site/@id will get id attributes of site elements, where site element is the child of current context node(day)

(c) xml_test(date) will convert raw text of date into proper R data type.

(d)

the textxml object is a list object with same length of days object.

Every element of this list is a data frame with different number of rows but with fixed 5 columns whose columns names are day, site, class, direction,count.

3.

(a)

Csv file is the smallest since raw data is only seperated by comma with a colunmns name in the first line. Json and XML are much larger since for every data element, it does not only contains raw data, but also has elemnt name and probality arrtibute for this data, and they are hierachical structures, which also need extra information to respresent.

(b)

The memory allocated during the call to read.csv() is about 2.35 Mb, while the memory of testcsv object is 0.447Mb. This is reasonable since in the process of creating a object, we need allocate more memory than the object itself to get this task done.

(c)

The difference between the output of the two calls to gc() for Vcells is that used(Mb) increases from 8.8 to 9.3. We notice that the object.size(testcsv) is 446416 bytes, around 0.446 Mb, which is exactly the difference of used memory(9.3 - 8.8).

(d)

The testcsv and test json are dataframe objects, with same data content and data type, hence their size are basically the same.

testxml, as being said, it is a list of dataframes. Although the total number of records are same the above two, but it has a different structure. Besides, all of its atomic data record is character, which may also cost larger memory than interger or numeric. Overall, it is three times larger that the above two objects.

4

(a)

The elapsed time is 'real' elapsed time since the code is processing.

The 'user time' is the CPU time charged for the execution of user instructions of the calling process.

For most of situations, elapsed time is longer since it is basically the sum of user time and system time.

(b)590 ms.

(c) The call of lapply

Since it is a implied loop which calls xml_find() and data.frame() and xml_test() functions multiple times

(d)

xml_find_all() (320ms) took up more time than xml_text(220ms)

(e) We many call xml_find_all one time by vectriztion instead of calling it in a loop.