1. What does sample() do, and what is each argument for?

This is the result I get when executing the sample() function in RStudio:

```
> sample(100, 10, replace = TRUE)

[1] 31 97 50 45  3 46 44  3 51 48
```

There are three clues for guessing what does the sample() function and each argument do:

a. Function name: the name "sample" resembles the sampling method we used to gather a subset of population for statistical analysis. We may guess that this sample() function also does something similar in getting sample data points.

b. Changing the 1st parameter value:

When the first argument is changed to 10, this is the result:

```
> sample(10, 10, replace = TRUE)

[1]  7  6  1 10 10 10  9 10 10  4
```

It seems that the 1st arguments specifies the upper range of each output number. The lower range is default to be 1 (the range is inclusive of boundary value). There seems to be error handling as well, in the sense that when the first argument is 0 or negative, all output is the first argument:

```
> sample(-3, 10, replace = TRUE)

[1] -3 -3 -3 -3 -3 -3 -3 -3 -3 -3
```

c. Changing the 2nd parameter value:

When the second argument is changed to 3, this is the result:

```
> sample(100, 3, replace = TRUE)
[1] 13 83 36
```

It is straightforward to guess that the 2nd argument indicates the number of data points in the output, i.e. the sample size.

It is confirmed in the error message too:

```
> sample(5, -1, replace = TRUE)
Error in sample.int(x, size, replace, prob) : invalid 'size' argument
```

d. Analyse error message:

The third argument is harder to guess. Thus, it helps to enter different values and examine both the output and the error message:

```
> sample(5, 10)

Error in sample.int(x, size, replace, prob) :

cannot take a sample larger than the population when 'replace = FALSE'
```

> sample(5, 3)

[1] 1 2 5

> sample(5, 10, replace=TRUE)

[1] 5 2 4 2 1 4 5 3 3 1

From the above three cases, it may be deduced that the 3rd argument specifies if it is possible to take sample from a population where population size is smaller than sample size.

A more detailed explanation is obtained from the "Help" section:

sample {base} R Documentation

Random Samples and Permutations

Description

`sample` takes a sample of the specified size from the elements of `x` using either with or without replacement.

Usage

```
sample(x, size, replace = FALSE, prob = NULL)

sample.int(n, size = n, replace = FALSE, prob = NULL)
```

Arguments

`x`	Either a vector of one or more elements from which to choose, or a positive integer. See 'Details.'
`n`	a positive number, the number of items to choose from. See 'Details.'
`size`	a non-negative integer giving the number of items to choose.
`replace`	Should sampling be with replacement?
`prob`	A vector of probability weights for obtaining the elements of the vector being sampled.

This documentation provides a more comprehensive guide from the implementer's point of view, but in my opinion, having too much information may do more harm than good sometimes. A simpler and equally effective way may be trying and examining the output as shown previously.

2. **How do the following functions work? str(); summary(); head(); tail()**

> str(acled)

'data.frame': 32291 obs. of 25 variables:

$ GWNO : int 615 615 615 615 615 615 615 615 615 615 ...

The function name “str” come from the first three letter of the word “structure”. The str() function displays the internal structure of the input object in a compacted manner.

When str(acled) is called, each field of the object is displayed in the structure of “$FIELD_NAME value value value …”. Note the ellipsis at the end of each field details that omits excessive information.

```
> summary(acled)
          GWNO       EVENT_ID_C        EVENT_ID_N
Min.   :404.0   Length:32291       Min.   :  3025
1st Qu.:490.0   Class :character   1st Qu.: 31562
Median :520.0   Mode  :character   Median : 60467
```

summary() is similar with str(), but their differences is important too. It summarises the input object not by showing a partial list of the entire entry, but by presenting the general statistics of the object, to provide not a sneak peek, but an overview.

When summary(acled) is called, the statistics such as minimum value, data length, and median are presented for each field.

```
> head(acled)
          GWNO EVENT_ID_C EVENT_ID_N EVENT_DATE YEAR TIME_PRECI
1  615    3030ALG      3025 2014-01-02 2014         1
2  615    3031ALG      3026 2014-01-03 2014         1
3  615    3032ALG      3027 2014-01-04 2014         1
```

The meaning of “head” in our common sense is the top portion of the body. The meaning of head() function is not dissimilar. It returns the top portion of the input object.

When head(acled) is called, the first portion of acled data set, starting from first index 1 is returned.

```
> tail(acled)
          GWNO EVENT_ID_C EVENT_ID_N EVENT_DATE YEAR
32286  552    5363ZIM     117818 2015-11-30 2015
32287  552    5364ZIM     117819 2015-12-05 2015
32288  552    5365ZIM     117820 2015-12-10 2015
```

The meaning of “tail” in our common sense is the end portion of the body. The meaning of tail() function is not dissimilar. It returns the last portion of the input object.

When tail(acled) is called, the last portion of acled data set, starting from ending index to the last index, is returned.

3. **How many events were there in 2014-2015 with more than 400 fatalities?**
   > acled[acled$FATALITIES > 400,] returns three rows of data:

Index	Fatalities
4630	420
14494	600
15909	500

   Thus, there were 3 events in 2014-2015 with more than 400 fatalities.

4. **What is the country with the largest number of events?**

   The commands I entered are as follows:

   ```
   > table(acled$COUNTRY)
   > sort(table(acled$COUNTRY))
   ```

   The last entry in the output is Somalia (5307 events), the country with the largest number of events.

5. **During which day of the week do most events happen?**

   The commands I entered are as follows:

   ```
   > acled$weekday <- weekdays(as.Date(acled$EVENT_DATE))
   > table(acled$weekday)
   > sort(table(acled$weekday))
   ```

   The last entry in the output is Wednesday (5416 events), the day of the week when the most number of events happened.

6. **How do the following functions work? substr(); aggregate()**

   substr()

   It has the structure substr(x, start, stop), and it is used as follows in the assignment:

   ```
   > acled$month <- substr(acled$EVENT_DATE, 1, 7)
   ```

   The data in $EVENT_DATE has the format of “2014-09-22”, a string of 10 characters. The substring with starting index 1 and ending index 7 preserves the first 7 characters, so the above becomes “2014-09”.

   aggregate()

   It is used in the assignment as follows, which serves as the x-variable in the line chart.

   ```
   aggregate(FATALITIES ~ month, acled, sum)
   ```

   “FATALITIES ~ month” is splitting the field FATALITIES into groups of months, “acled” is the name of the dataset, and “sum” is the feature to be summarised. Thus, aggregate() splits the data “acled”

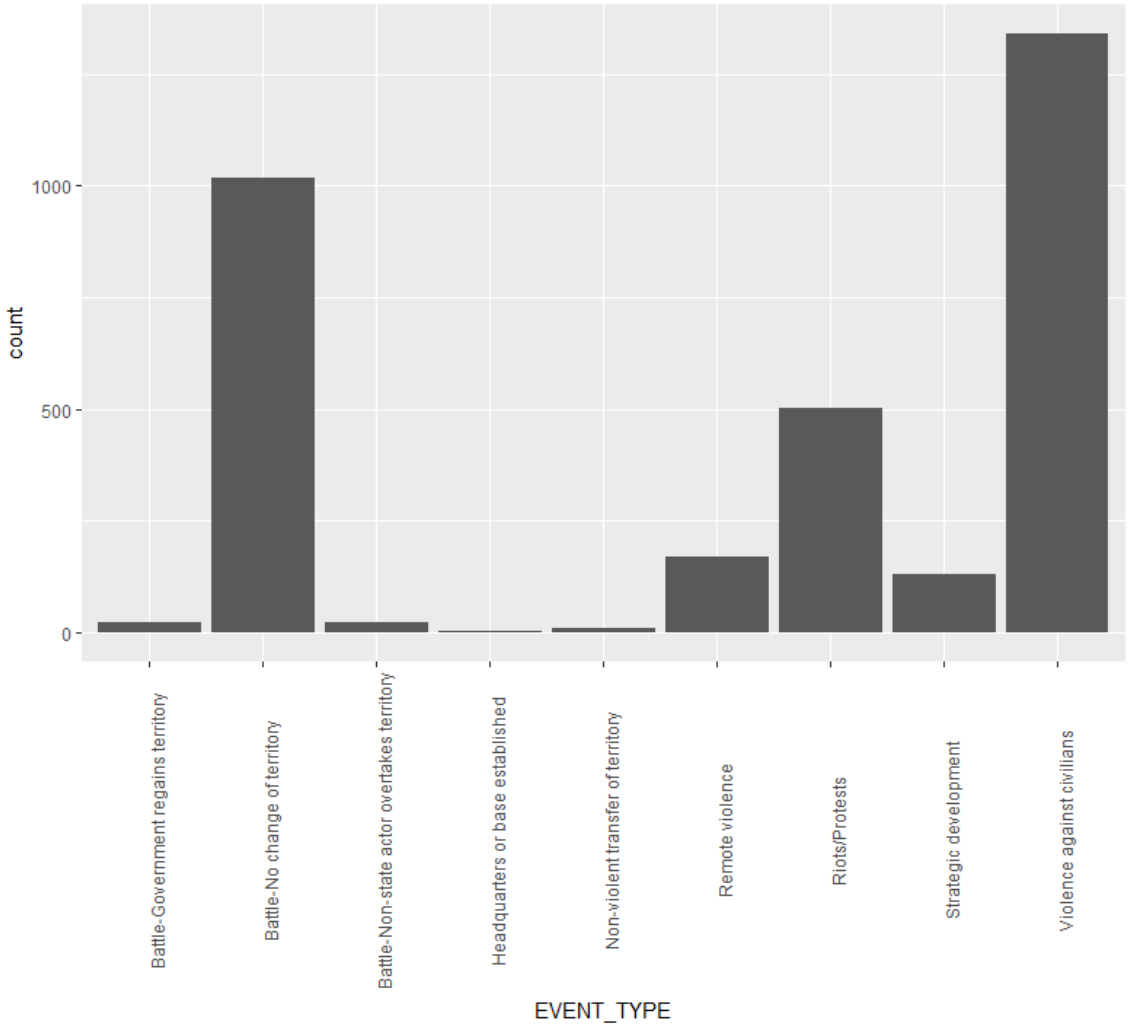
into subsets of “FATALITIES ~ months”, computes summary statistics “sum” for each, and returns the result in a convenient form.

7. **What was the most frequent event type in 2004?**

   The command I entered was:

   > ggplot(acled[acled$YEAR == 2004,], aes(EVENT_TYPE)) + geom_bar()

   From the following bar chart, the most frequent event type in 2004 is Riots/Protests.



8. **Which year had the most fatalities due to violent events against civilians?**

   The command I entered was:

   acled$violenceAgainstCivilians<- acled$EVENT_TYPE=='Violence against civilians'

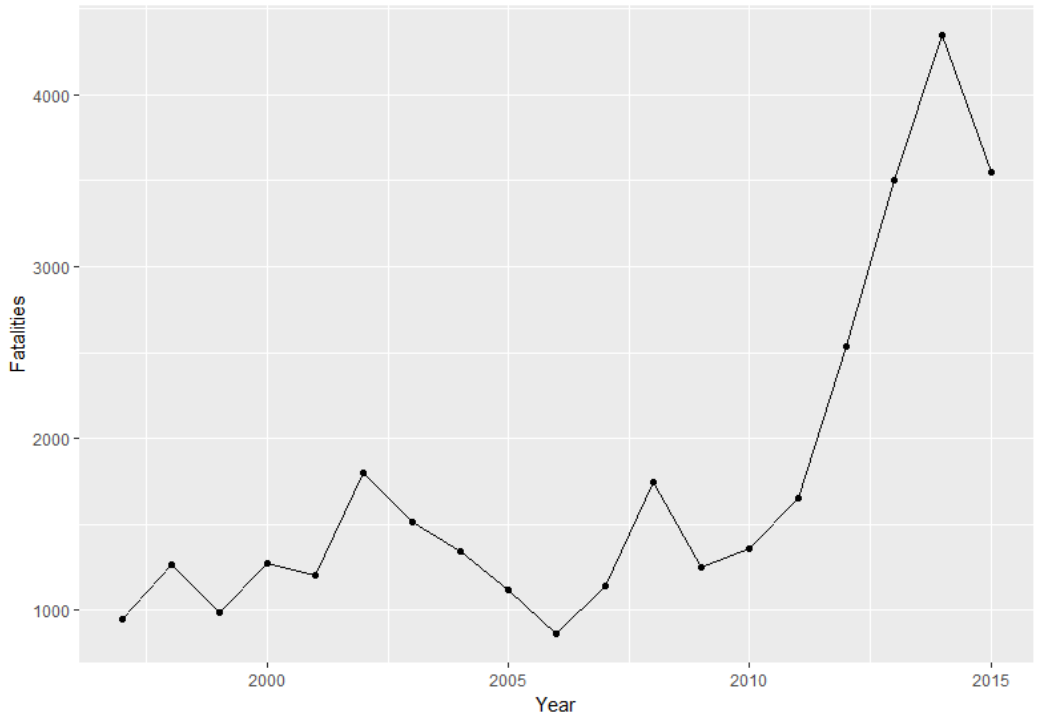
   violenceAgainstCiviliansSumPerYear <- aggregate(violenceAgainstCivilians ~ acled$YEAR, acled, sum)

This “violenceAgainstCiviliansSumPerYear” stores the following table:

```
> violenceAgainstCiviliansSumPerYear
   acled$YEAR violenceAgainstCivilians
1        1997                      950
2        1998                     1267
3        1999                      986
4        2000                     1274
5        2001                     1207
6        2002                     1801
7        2003                     1515
8        2004                     1340
9        2005                     1117
10       2006                      865
11       2007                     1141
12       2008                     1745
13       2009                     1250
14       2010                     1355
15       2011                     1649
16       2012                     2536
17       2013                     3503
18       2014                     4343
19       2015                     3549
```

ggplot(violenceAgainstCiviliansSumPerYear, aes(acled$YEAR, violenceAgainstCivilians, group = 1)) + geom_line() + geom_point()

From the following line chart, the year with the most fatalities due to violent events against civilians is 2014.



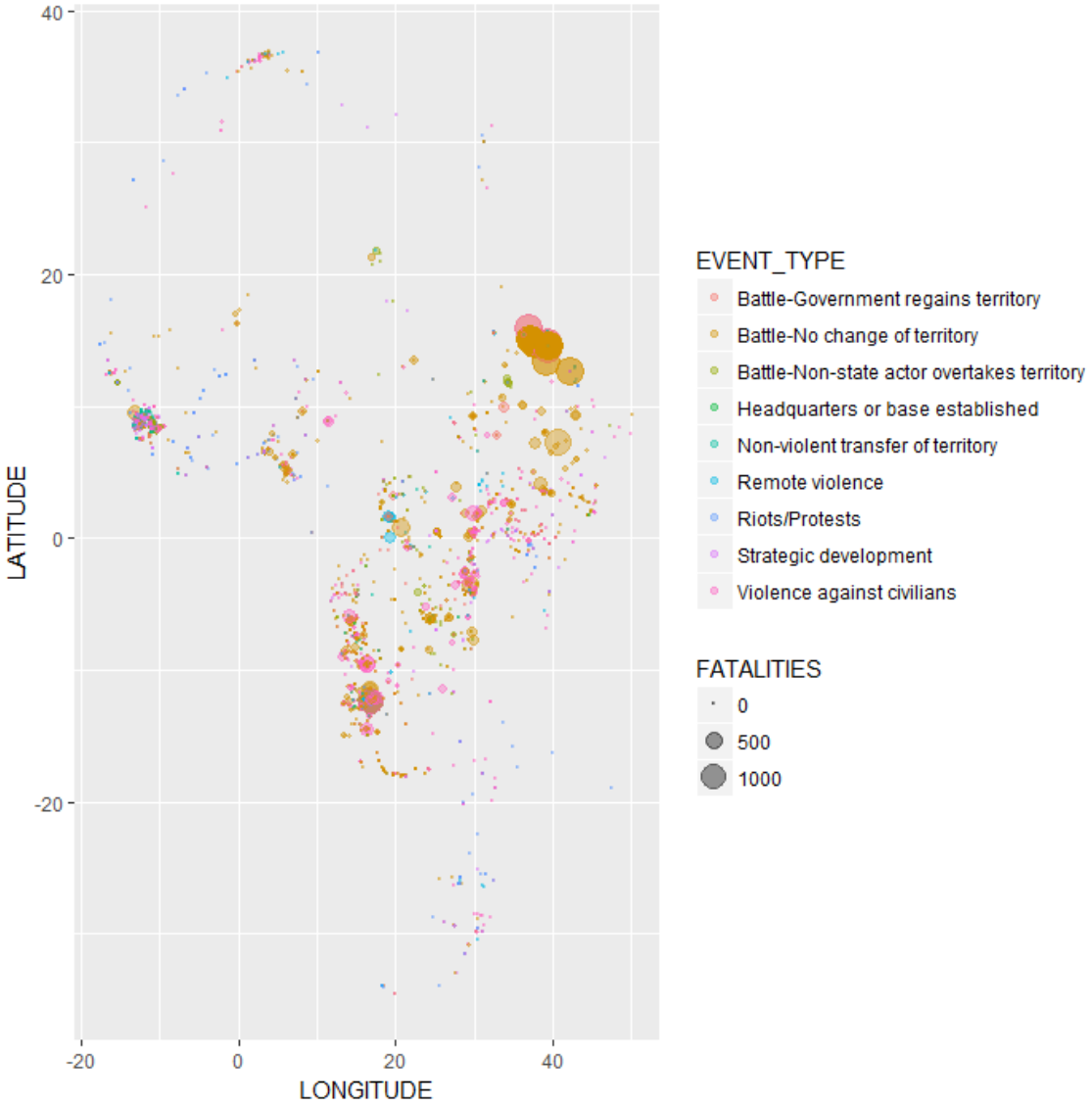
9. Create two proportional symbol maps showing the location and number of fatalities of events in 1999 and 2004. Embed them in your assignment

The command I entered was:

> ggplot(acled[acled$YEAR == 1999,], aes(y = LATITUDE, x = LONGITUDE))+
geom_point(alpha=0.4,aes(size = FATALITIES, colour=EVENT_TYPE))+scale_size_area()

> ggplot(acled[acled$YEAR == 2004,], aes(y = LATITUDE, x = LONGITUDE))+
geom_point(alpha=0.4,aes(size = FATALITIES, colour=EVENT_TYPE))+scale_size_area()

The proportional symbol map for showing the location and number of fatalities of events in 1999:



The proportional symbol map for showing the location and number of fatalities of events in 2004: