MTH 3210 Lab 1

Due Thu., Feb. 7

1 Part A: Graphing and Summarizing Data

1.1 Lightning Deaths dataset

The following data represent numbers of deaths by lightning strikes in the U.S. for each of the years 1959 - 2005 (in time order), as compiled by the National Climatic Data Center from reports by the National Weather Service.

75,	48,	61,	48,	150,	49,	57,	39,	27,	51,	46,	50,	62,	51,	50,	58,
38,	34,	59,	44,	24,	39,	40,	33,	49,	33,	34,	32,	35,	30,	23,	39,
36,	25,	20,	32,	43,	52,	42,	44,	46,	51,	47,	51,	44,	33,	38	

1. The "combine" function `c()` combines data values into a univariate dataset (called a ***vector*** in R). The assignment operator `<-` is used to assign a name to the dataset. For example,

```
myData <- c(3, 7, 5, 9, 4)
```

   stores the dataset 3, 7, 5, 9, 4 into `myData`, a *vector*. We can view a dataset by typing its name:

```
myData

## [1] 3 7 5 9 4
```

   Use the `c()` function and the assignment operator `<-` to create a vector containing the **lightning deaths** data.

2. The function `hist()` will produce a histogram a dataset. The main argument passed to `hist()` is a data vector `x`, example, typing

```
hist(x = deaths)
```

   will make a histogram of the **lightning deaths** data (assuming you named your data vector `deaths`).

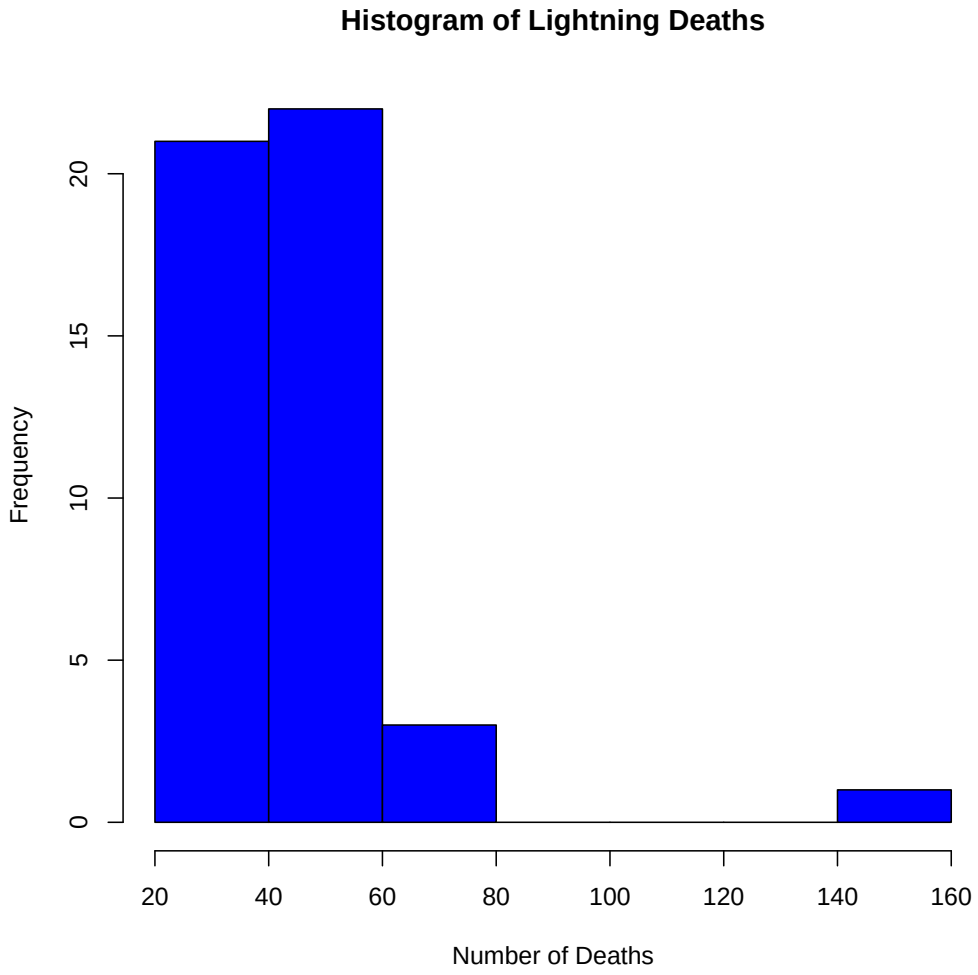
   But `hist()` accepts other optional arguments too. Among its other arguments are:

`x`	a data vector.
`col`	a color used to fill the histogram bars.
`xlab`	a label for the x-axis.
`ylab`	a label for the y-axis.
`main`	a main title.

Use `hist()` to make a histogram of the **lightning deaths** data, for example by typing something like this (assuming you named your data vector `deaths`):

```
hist(x = deaths, col = "blue", xlab = "Number of Deaths", ylab = "Frequency",
       main = "Histogram of Deaths Due to Lightning")
```

Your histogram should look something like the one below.



3. The function `boxplot()` will produce a boxplot of a dataset. The main argument passed to `boxplot()` is a data vector `x`, but it accepts other optional arguments too. Among its arguments are:

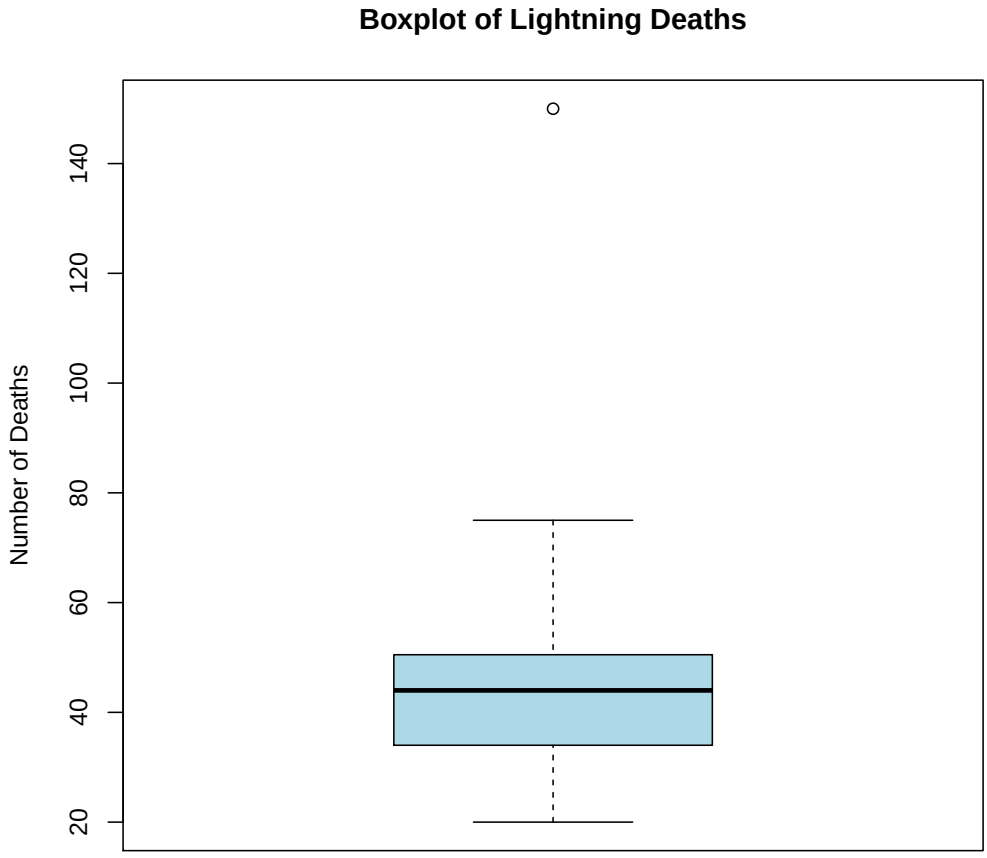
`x`	a data vector.
`col`	a color used to fill the body of the box.
`xlab`	a label for the x-axis.
`ylab`	a label for the y-axis.

`main`	a main title.
`names`	a character vector of group names for the x-axis.

Make a boxplot of the **lightning deaths** data, for example by typing something like this:

```
boxplot(x = deaths,  col = "lightblue", main = "Boxplot of Lightning Deaths",
          ylab = "Number of Deaths")
```

Your plot should look something like the one below.



4. The function `length()` will count how many elements (observations) are in a data vector. Use `length()` to determine how many observations there are in the **lightning deaths** dataset.

5. Use the function `sum()` to count the **total** number of deaths over the years 1959 - 2005.

6. Use the functions `min()` and `max()` to find the minimum and maximum values in the dataset.

7. Use `mean()` and `median()` to compute the mean and median, and compare their values.

8. Use `sd()` and `IQR()` to compute the standard deviation and interquartile range of the data.

9. Regarding the outlier (150 deaths in 1963), a National Weather Service report states:

   > "On December 8, 1963 the crash of a jetliner killing 81 people near Elkin, Maryland, was attributed to lightning by the Civil Aeronautics Board investigators."

   We can use square bracket operator `[ ]` to access specific values in a dataset. For example, typing:

   ```
   deaths[5]

   ## [1] 150
   ```

   returns the 5th value from `deaths`.

   We can also use square brackets to *replace* a value in a dataset.

   Use square brackets `[ ]` to replace the death count of 1963 (the 5th value in the dataset) by 69 (= 150 - 81) in the **lightning deaths** dataset by typing:

   ```
   deaths[5] <- 69
   ```

10. Now, after having replaced the outlier by a more legitimate value (Step 9), recompute the mean, median, standard deviation and interquartile range and compare your answers to those of Steps 7 and 8.

2 Part B: Linear Transformations of Data Values

2.1 Lake Tahoe and Donner Lake Temperature dataset

The following data are water temperature measurements degrees Celsius (°C) made in 2001 at $n = 9$ sites on Lake Tahoe and Donner Lake, near the border of California and Nevada.

Site Name	Temperature
Tahoe City	17.5
TRG Buoy	17.5
Sugar Pine Point	17.8
Emerald Bay	18.3
Ski Run Marina	18.8
Camp Richardson	17.8
Tahoe Keys Marina East	19.3
Donner Lake Boat Ramp	20.3
Donner Lake Buoy	20.0

Here are the data again in a more convenient format:

$$17.5,\ 17.5,\ 17.8,\ 18.3,\ 18.8,\ 17.8,\ 19.3,\ 20.3,\ 20.0$$

1. Use `c()` and `<-` to create a vector containing the data.
2. Use `mean()` and `sd()` to compute the mean and standard deviation.
3. Recall that if $x_1, x_2, \ldots, x_n$ is a dataset and we perform a linear transformation

   $$y_i = ax_i + b$$

   for each $i = 1, 2, \ldots, n$, then the new mean and standard deviation are related to the old ones by

   $$\bar{y} = a\bar{x} + b \qquad \text{and} \qquad s_y = |\,a\,|s_x \tag{1}$$

   To convert a temperature in degrees Celsius (°C) to one in degrees Fahrenheit (°F), we use:

   $$^\circ\text{F} = \frac{9}{5}\,^\circ\text{C} + 32$$

   Use (1) and your $\bar{x}$ and s_x values from Step 2 to determine what the new mean and standard deviation would be if we were to convert the Lake Tahoe and Donner Lake temperature data to (°F).
4. R performs computations on vectors ***elementwise*** (one value at a time). For example, if `x` is a dataset containing temperatures in °C, then typing

   ```
   y <- 9/5 * x + 32
   ```

   converts each value in `x` to °F one at a time and stores the results in `y`.

   Create a new vector containing the Lake Tahoe and Donner Lake temperatures in °F.
5. Use `mean()` and `sd()` to compute the mean and standard deviation of the °F temperature data and compare their values to your answers from Step 3.

3 Part C: Side-by-Side Boxplots

3.1 Blood Coagulation Times Data

The table below shows coagulation times (seconds) for samples of blood drawn from 24 animals receiving four different diets, A, B, C, and D. The animals were randomly assigned to the diets, and the blood samples were drawn and tested in random order.

Blood Coagulation Times

Diet A	Diet B	Diet C	Diet D
62	63	68	56
60	67	66	62
63	71	71	60
59	64	67	61
59	65	68	63
63	66	68	64
62	64	66	63
61	67	68	59

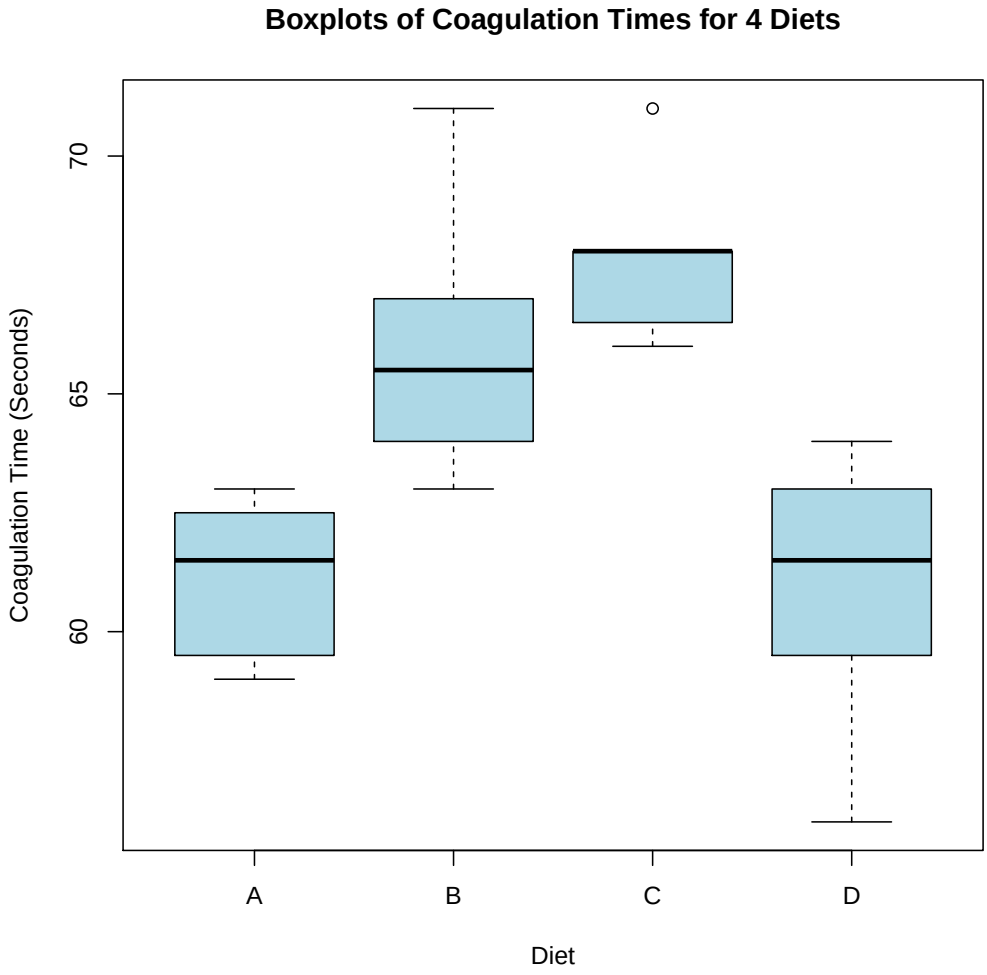
1. Read the data into four vectors using the following R commands:

```
DietA <- c(62, 60, 63, 59, 59, 63, 62, 61)
DietB <- c(63, 67, 71, 64, 65, 66, 64, 67)
DietC <- c(68, 66, 71, 67, 68, 68, 66, 68)
DietD <- c(56, 62, 60, 61, 63, 64, 63, 59)
```

2. The `boxplot()` function takes any number of vectors as arguments and plots them as side-by-side boxplots (e.g. `boxplot(x, y, z)` plots the vectors `x`, `y`, and `z`).

   Use `boxplot()` to make side-by-side boxplots of the blood coagulation times for the four diets. (Refer to the list of optional arguments that can be passed to `boxplot()` in Step 3 of Part A.)

   The result should look something like this:


Boxplots of Coagulation Times for 4 Diets
Coagulation Time (Seconds)
60
65
70
A
B
C
D
Diet