Simulating Confidence Intervals for Mean and Variance using Real Data in R Programming Environment

Leslie Chandrakantha
lchandra@jjay.cuny.edu
Department of Mathematics & Computer Science
John Jay College of Criminal Justice of CUNY
USA

ABSTRACT

Research has shown that the use of computer simulation methods as an alternative to traditional methods enhances the understanding of the statistical concepts. The increasing availability of technology allows instructors and students to use computationally intensive methods such as simulation. This paper presents the use of real data and a simulation approach to help students understand the confidence intervals for population mean and variance. Use of real data makes the concepts more real for students and enhances their ability to ground the new concepts in their existing knowledge. We use the R programming environment for simulating repeated sampling from a fairly large dataset and compute the approximate sampling distributions of sample mean and variance. We notice that the confidence intervals for population variance work poorly if the normality assumption is violated.

Introduction

Statistics is used in most of our everyday activities and it is applicable in a wide variety of settings such as in crime reports, weather reports, and education. This wide applicability of statistics in many areas makes it essential for students to have a good understanding of the concepts. Understanding fundamental statistical concepts in introductory level classes is a challenge for students. Many students struggle to grasp the inferential statistics concepts in sampling distribution, confidence intervals, and hypothesis testing. Due to the abstract nature of these topics, traditional way of teaching using books and lecture based instructions does not give a good understanding of the concepts to many students. We feel this difficulty can be confounded by introducing concepts through a focus on abstract mathematical models. According to the American Statistical Association's *Guidelines for Assessment and Instruction in Statistics Education* (GAISE) [8], an introductory statistics course should: promote statistical literacy and statistical thinking, use real data, promote conceptual understanding, foster active learning, use technology for developing conceptual understanding and analyzing data, use assessments to improve and evaluate student learning. This paper will focus on the second and fifth guidelines, the use of real data, and the use of technology in teaching statistics.

In this paper, we describe the use of real data and computer simulation to introduce statistical concepts. The use of real data in teaching statistical concepts has been increasingly recommended in statistics education. Schaeffer [15] noted that real-life data should be used when teaching statistics, particularly data that is of interest and relevance to the lives and experience of students. In 2005, the American Statistical Association's GAISE report (noted in the previous paragraph) included the recommendation to use real data to achieve learning goals given by Franklin & Garfield [7]. According to Conner & Davis [4], the national curriculum guidelines for teaching statistics in many countries including Australia and England emphasized the need to use real data. Garfield & Ben-Zvi [9] noted that students will construct knowledge based on their experiences using real data sets. Willett & Singer [16] have recommended instructors and textbook authors to use real data. They pointed out that students

learn skills in a realistic and relevant context. In addition to being more interesting, real data sets provide a practical arena in which students can learn how to link research questions to statistical models. Real data gives students a chance to reflect upon their work with the data while instructors provide guidance to build their own understanding. Pfannkuch [14] noted that students may develop their statistical reasoning through an interaction between their contextual knowledge about the data set and their emerging statistical knowledge. Students can use data sets to practice calculations, gain experience in the interpretation of the results, and develop their statistical reasoning about a problem. Instructors may also use data sets to introduce different research approaches, methods of data analysis, and applications of statistical theory to solve real life problems.

The second GAISE report guideline we focus on in this paper is the use of technology in teaching statistics. Technology strongly influences how we teach statistics. Advances in technology have enabled instructors to experiment with different teaching methods. Today, computers and software, calculators, and graphing calculators allow users to perform many functions that may not be possible without them. Technology turns statistics into a hands-on activity-based course where students become engaged in statistics by not just learning about formulas. Ang[1] noted that technology and technology based environments allow students to alleviate some of the difficulties of grasping the concepts in mathematical modeling and in related disciplines. Simulation is the imitation of the operations of a real life process or system. Simulation can be an effective tool in learning abstract concepts in statistics. Technology allows students to conduct simulations which let them experience the long term behavior of sample statistics under repeated random sampling. Through simulations, students can build on their intuitions about probability and expected values, and come to understand the behavior of sampling distributions in order to explore the patterns inherent in randomness. Mills [13] has given a comprehensive review of the literature of computer simulation methods used in all areas of statistics to help students understand difficult concepts.

R is a free, powerful, and flexible statistical programming language and computing environment that has become very popular among statisticians. R runs on all of the commonly used computer platforms including Windows, Unix/Linux, and the Macintosh operating system. Many introductory and higher level statistics instructors are now using R to teach and perform statistical data analysis. Although it is an initial challenge for students to write statements in the command line, R can be used to conduct data analysis effectively. R can easily generate random samples from many data sets and a variety of probability distributions. Hallgren [12] has used R in data analysis, estimating statistical power, and constructing confidence intervals of parameters. He noted that simulation methods are flexible and can be applied to a number of problems to obtain answers that may not be possible to derive through other approaches.

In this paper, we describe the use of real data sets and R in an introductory statistics class to understand the concepts of confidence intervals for population mean and variance. We generate random samples from a fairly large real data set and compute confidence intervals. R syntax is used to repeat this process for a large number of times to understand the meaning of a confidence level. We observe the effects of non-normality of the population data on the confidence intervals.

In section 2, we give quick tour on R. Section 3 gives a description of the real data set. In sections 4 and 5, we demonstrate the simulation of confidence intervals for mean and variance. We end the paper in section 6 with some concluding remarks.

A Quick Tour of R

Although it is not feasible to provide a general introduction to R in this paper, we will provide enough background information to understand the remainder of the paper. R is a relatively simple syntax-driven and case-sensitive language. Even though the syntax for writing instructions may be somewhat difficult initially, most students with little or no prior programming experience have become comfortable using R. R is an object-oriented program that works with data structures such as vectors (one-dimensional array) and data frames (two-dimensional arrays). A vector contains a list of values. When the R program is started and after it prints an introductory message, the R interpreter prompts for input with > (the greater-than sign). The interpreter executes expressions that are typed at the command prompt. For example:

```
> 1 + 3*4
[1] 13
> 1:4
[1] 1 2 3 4
> 1:4 + c(2,6,-3,4)
[1] 3 8 0 8
> x <- 1:4
> 3*x
[1]  3  6  9 12
```

Most of the above R statements are self-explanatory except the following:

- Simple (vector) output is prefixed by [1]. If the output extends over several lines , the index number of the first element in each line appears in square brackets at the beginning of the line
- c() function combines its arguments to create a vector. The arguments are specified within parenthesis and separated by commas.
- <- is the assignment operator. The equal sign (=) may also be used for assignment purposes. Variables are created and memory is allocated to them dynamically. Variable names can consist of any combination of lower and upper case letters, numerals, periods, and underscores, but cannot begin with a numeral or an underscore. R is case sensitive and there is no limit on the number of characters in a name.

Once we have a vector of numbers, we can apply built-in functions to get useful statistical summaries and visual displays. R also provides functions for generating random samples from various probability distributions.

***sample* and *subset* functions**

The *sample* function generates a random sample of specified size from a set of values with or without replacement. Let's suppose values are stored in a vector named x. To take a random sample of size *n* without replacement from the set x, we use following R commands:

> *sample(x, n)* or > *sample(x, n, replace = FALSE)*

To obtain a sample of size *n* with replacement, we use following command:

> sample(x, n, replace = TRUE)
We can use the *sample* function to obtain a random number from a set of numbers, say 1 through 10, in following way:

> sample(1:10,1)
[1] 2

The *subset* function returns a subset of a vector or data frame which meets a particular condition. The following command returns the values in vector x that are greater than zero:

> subset(x, x > 0)

Data Input
Variables with small data sets can be directly entered at the keyboard, but this approach is limited. R has many other ways to input data. Data can be read from text files, csv (comma-separated values) files, and attached packages.

read.table function is used to import data from a text file. Similarly, *read.csv* function imports data from a csv file.

To read data into a data frame named *mydata* from a text file named *values.txt* resides in c:\data\values.txt:

> mydata <- read.table("c:\data\values.txt", header = TRUE)

The first line of the file should have a name for each variable in the data frame. However, if the first row does not contain names of variables then header argument should be set to FALSE.
To read from a csv file, replace *read.table* with *read.csv*.

In addition, you can read in files using the *file.choose()* function in R. After typing the following command in R, you can manually select the directory and the file where your dataset is located.

> mydata <- read.table(file.choose(), header = TRUE)

The *attach* function can be used to make objects contained in data frames accessible. The following command allows the user to access data in *mydata* data frame:

> attach(mydata)

Control Structures
R has the standard control structures such as *if, while,* and *for*. These can be used to control the flow of an R code. We will demonstrate the use of control structures in R using the following code segment. Let's assume that we have stored 1000 numbers in the vector named x. The following code will compute the average of the nonnegative numbers in vector x. The symbol # is used to write comments.

```
> total <- 0          # variable total initialized to 0
> count <- 0          # variable count initialized to 0
> for(i in 1:1000)
+ {
+   if (x[i] >= 0)
```

```
+    {
+     count <- count +1    # count the positive values
+     total <- total + x[i]    # add the positive values
+    }
+ }
> average <- total/count  # compute the average
```

In the above code segment, the *for* loop iterates 1000 times, selecting only nonnegative numbers using the *if* statement. It computes the average as well. The variable named *count* counts the number of nonnegative numbers stored in x. We will use the control structures when we discuss simulations in the following sections.

Real Data

Data used in this paper came from the General Social Survey (GSS) data set of year 2016. This survey is a sociological survey created and regularly collected since 1972 by the National Opinion Research Center at the University of Chicago. Since 1972, GSS has provided politicians, policymakers and scholars with a clear and unbiased perspective on what Americans think and feel about many social issues. GSS data are used in numerous newspaper, magazine, and journal articles. The GSS is also a major teaching tool in colleges and universities. More than 27,000 journal articles, books, and Ph.D. dissertations are based on the GSS and about 400,000 students use the GSS in their classes each year [11]. GSS data is freely available to interested parties over the internet. The data is generally available in formats designed for statistical programs such as R, SAS, and SPSS.

For the purpose of this paper, we use the data collected in year 2016 which is the latest data available. The 2016 data file has 2867 cases and 960 variables. We use the *hrs1* variable which represents the number of hours worked last week by the respondents of the survey for our data analysis. There were three types of missing values in the data. They are inapplicable (IAP), do not know (DK), and no answer (NA). The numbers -1, 98, and, 99 respectively, were used to indicate these missing values in the data set. We eliminate these missing values in following way. First, we save *hrs1* variable data into a csv file named *hrs1data.csv*. This file has a column heading hrs1. Then we import this file using following command:

```
> file1 <- read.csv(file.choose(), header = TRUE)
```

After selecting the *hrs1data.csv* file from the directory, we make the data accessible to R by using the *attach* function.

```
> attach(file1)
```

Now we eliminate missing values from the *hrs1* data and assign it to a new variable named *hrs2* in the following command:

```
> hrs2 <- subset(hrs1, hrs1 !=-1 & hrs1 != 98 & hrs1 != 99)
```

In the above subset function, the logical condition *hrs1 !=-1 & hrs1 != 98 & hrs1 != 99* selects data values that are not equal to -1, 98, and 99 and assign them to the *hrs2* variable. The *hrs2* variable has 1646 data values. This will represent the population for the purpose of this paper. We will use *hrs2* in coming sections to study the concepts of confidence intervals.

Simulating Confidence Intervals for population mean (μ)

Confidence intervals are one of the most commonly used statistical methods to estimate plausible values of population parameters using sample data. However, both the formal concept and the intuition behind confidence intervals remain elusive to many students. Confidence intervals give the most likely range of the unknown population parameter. Garfield, delMas, & Chance [10] have listed the following points for confidence intervals that students should understand:

- A confidence interval for a population mean is an interval estimate of an unknown population parameter (mean) based on a random sample from population.
- A confidence interval for a population mean is a set of plausible values of the true population mean that could have generated the observed data as a likely outcome.
- The level of confidence tells the probability that the method produced an interval that includes the true population parameter.

In this section, we consider creating and interpreting the confidence interval for the mean (μ) assuming that the population standard deviation (σ) is known by using simulation. A random sample of size *n* is taken from the population of data. For sufficiently larger *n*, the Central Limit Theorem (CLT) implies that the sample mean $\bar{X}$ has approximately a normal distribution regardless of the nature of the population distribution. The larger the sample size, the better the normal approximation. It then follows that $z = (\bar{X} - \mu)/(\sigma/\sqrt{n})$ has approximately a standard normal distribution. Generally speaking, n > 30 will be sufficient for the normal approximation for sample mean. However if the original population is more skewed, a larger sample is needed to use the normal approximation. See Chandrakantha [3] for more details. Then it can be derived that the confidence interval for mean μ is $\bar{X} \pm Z_{\alpha/2}\sigma/\sqrt{n}$, where $Z_{\alpha/2}$ is the value of the standard normal curve with area (1-α) between critical points $-Z_{\alpha/2}$ and $Z_{\alpha/2}$, *n* is the sample size. *The confidence level (1-α) is the probability that the confidence interval actually does contain the population mean μ, assuming the estimation process is repeated a large number of times.* Students have major difficulties in understanding this last statement. Fidler [6] has noted the following misconceptions of student understanding of confidence intervals: the majority of the individual values are in the interval, the interval contains the plausible values of the sample mean, covers 100(1-α)% of the sample, and the probability that the population mean is contained within a level (1-α) confidence interval is (1-α). It is important that students understand that in repeated sampling from a population, 100(1-α)% of intervals (say 95%) would capture the true unknown mean. In using the traditional way of teaching, we only consider one sample and calculate one interval. This leads them to believe the wrong interpretation of the interval, that there is a 95% chance that this interval will have the true mean. The iteration (or simulation) process builds a distribution of intervals, and can be displayed graphically in R.

A computer simulation method using R will allow students to understand the true meaning of the confidence interval. We use the real data set in the *hrs2* variable (mentioned in previous section) for this purpose. This data set represents the number of hours worked in the previous week by the respondents of the survey and it serves as our population or sampling frame. We generate many samples from this population and compute 95% confidence intervals for the mean number of hours worked. The confidence interval formula given in the previous paragraph is valid when sampling distribution of the sample mean is normally or approximately normally distributed. First, we verify this requirement. To do so, we study the population distribution of *hrs2* using the *hist(hrs2)* and *qqnorm(hrs2)* commands in R. *hist* and *qqnorm*

functions create histogram and normal probability plot, respectively. *Figure 1* shows both plots below:

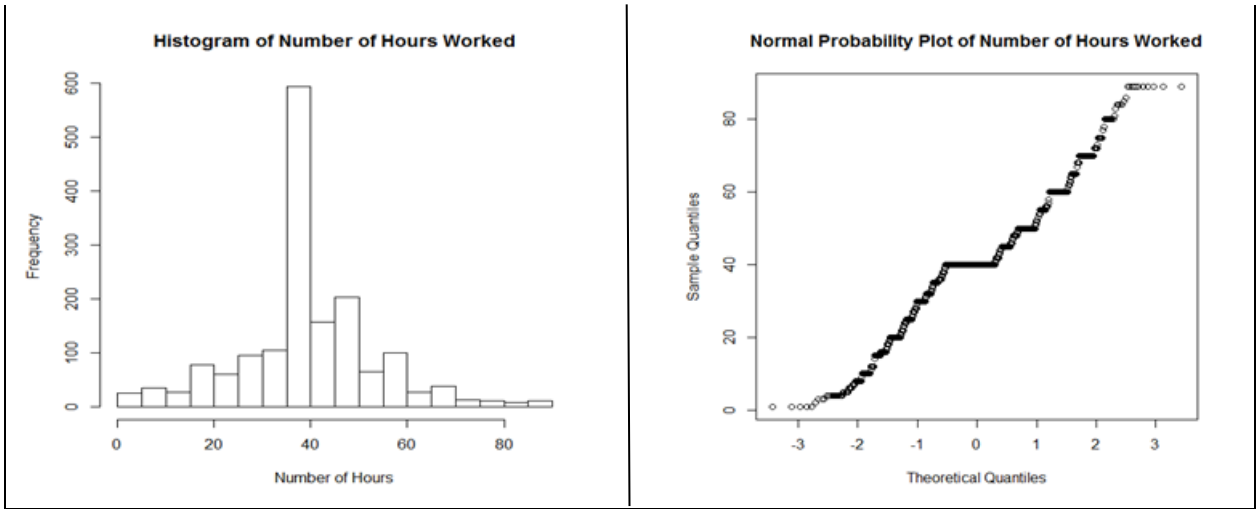

Figure 1: Histogram and Normal Probability Plot of Population Data

From both plots we notice that the data is not normally distributed. Significantly more hours spent working are between 35 and 40. According to the central limit theorem, if we take larger samples, the sampling distribution of the mean will be approximately normally distributed. With the sample size (n) of 100, we simulate the process of computing the sampling distribution of sample mean, $\bar{X}$. We use the following R code to select 1000 random samples of size 100 from the population, compute the sample mean, and create the histogram, normal probability plot, and box plot to verify the normality of the sampling distribution of the mean.

```
> mean_v <- NULL
> for(i in 1:1000)
   {
+    x <- sample(hrs2,100, replace = FALSE)
+   m <- mean(x)
+   mean_v <- c(mean_v,m)
   }
> hist(mean_v, breaks = 15, main  = "Histogram of Sample Means", xlab = "Sample Mean")
> qqnorm(mean_v, main = "Normal Probability Plot of Sample Means")
> boxplot(mean_v, main = "Boxplot of Sample Means", ylab = "Sample Mean")
```

In this code segment, we first create an empty (null) vector named *mean_v* to hold the sample means. The *for* loop repeats the process of taking 1000 random samples of size 100 without replacement, computes the sample means and saves them in the *mean_v* vector. Variable *x* is a vector that holds the random sample for each iteration of the *for* loop. *hist*, *qqnorm*, and *boxplot* functions draw the corresponding plot. The *Figure 2* shows these plots below. All three plots in *Figure 2* show that the sample means are approximately normally distributed and that they can be used to compute the confidence intervals.

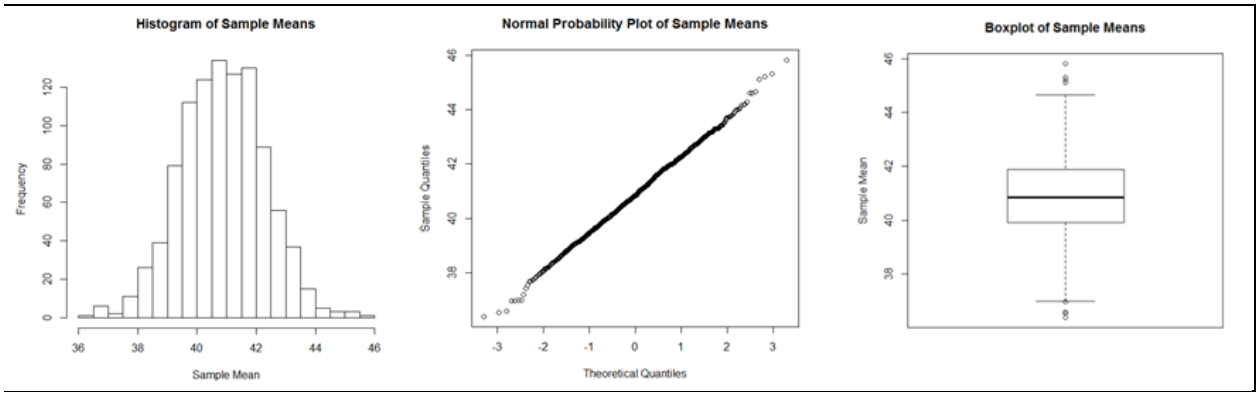

Figure 2: Histogram, Normal Probability Plot, and Boxplot of Sample Means

The end points of the 95% confidence interval are estimates of the 2.5th and 97.5th percentiles of the distribution of $\overline{X}$, Hallgren [12]. 95% confidence interval limits are obtained using the *quantile* function in the following R code segment. The *quantile* function computes the sample percentiles for given probabilities. For a 95% confidence level, we identify the values at the 2.5th and 97.5th percentiles of the *mean_*v vector (these values could be adjusted to obtain confidence interval limits for different confidence levels). The result is saved in a vector named *conflim*. Calling the *conflim* vector returns the values that correspond with 2.5th and 97.5th percentile of the empirical sampling distribution of mean.

```
> conflim <- c(quantile(mean_v,0.025), quantile(mean_v,0.975))
> conflim
  2.5%   97.5%
38.1400 43.5815
```

Based on the confidence interval computed above, we are 95% confident that the mean number of hours worked per week by the population is between 38.14 hours and 43.58 hours. The level of confidence tells the probability the method produced an interval that includes the true population parameter. To get a better understanding of this concept, we generate 1000 random samples from population data and compute a confidence interval for each sample using the confidence interval formula. For a 95% confidence level, this formula is $\overline{X} \pm 1.96 * \sigma / \sqrt{n}$. Using these 1000 confidence intervals, we find the proportion of those intervals containing the true mean which is known as the coverage probability. This proportion should be 95% which is the assumed confidence level when the population distribution is normal or close to it for larger sample sizes. The following R code simulates this process:

```
> lower_lim <- NULL
> upper_lim <- NULL
> for(i in 1:1000)
  {
+    x <- sample(hrs2,100, replace = FALSE)
+    l_lim <- mean(x) - 1.96*sd(hrs2)/sqrt(100)
+   u_lim <- mean(x) + 1.96*sd(hrs2)/sqrt(100)
+   lower_lim <- c(lower_lim,l_lim)
+   upper_lim <- c(upper_lim,u_lim)
 }
```

```
> count <- 0
> for(i in 1:1000)
  {
+     if(lower_lim[i] < mean(hrs2) & upper_lim[i] > mean(hrs2))
+         count <- count +1
  }
> count/1000
[1] 0.957
```

In this code segment, first we create two empty vectors named *lower_lim* and *upper_lim* to hold the confidence interval limits for 1000 simulated samples. In each iteration of the *for* loop, a random sample of size 100 selected without replacement from population data and assigned to variable *x*. The *sd* function computes the standard deviation of the population data. Using each sample, lower and upper limits of the confidence interval are calculated and saved in *lower_lim* and *upper_lim* vectors. After completion of the *for* loop, *lower_lim* and *upper_lim* vectors will contain 1000 confidence interval limits. Next *for* loop counts how many intervals would contain the true mean of the population data. In each iteration, the *count* variable will increment by one if the interval contains the mean. Dividing the *count* variable by 1000 at the end of the *for* loop will produce the proportion of intervals containing the population mean. In this simulation run, this proportion is 0.957 (95.7%). This means that in the long run, 95% of the computed confidence intervals will contain the population mean. To see this visually, we use the following R commands to draw 100 confidence intervals horizontally and draw a vertical line for the true mean. The resulting 100 confidence intervals are shown in *Figure 3*. It can be seen that 5 intervals do not contain the true mean indicating that 95% of the intervals contain the mean.

```
> matplot(rbind(lower_lim, upper_lim), rbind(1:100, 1:100), type = "l", lty = 1, xlab =
"Sample Means",ylab = "Samples", main = "Confidence Intervals for 100 Samples")
> abline(v = mean(hrs2))
```

The *matplot* function plots columns of one matrix against columns of another. In this case it plots the columns of (*1:100, 1:100*) matrix against columns of (*lower_lim, upper_lim*) matrix to produce the plot in *Figure 3*. *type = "l"* is used to draw lines between points and *lty = 1* draws solid lines. The *abline* function with *v = mean(hrs2)* draws a vertical line through the true mean.

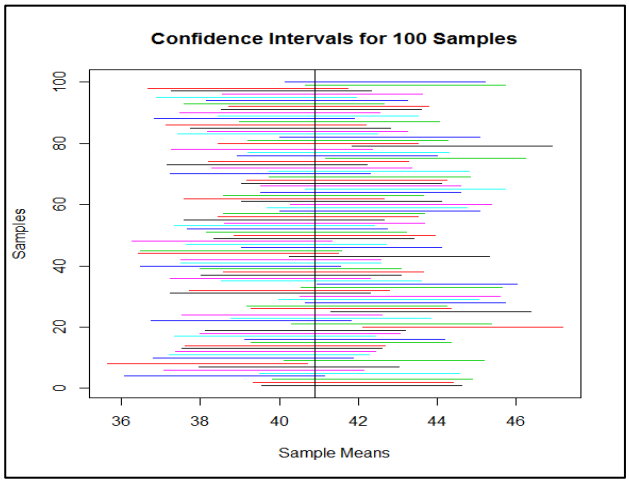

Figure 3: One Hundred 95% Confidence Intervals

To verify the accuracy of the 95% confidence interval calculated earlier using the 2.5th and 97.5th percentiles from empirical sampling distribution of the mean, we find the average of all 1000 intervals calculated using the confidence interval formula. The interval computed using these percentiles was (38.14, 43.58). To find the average of 1000 confidence interval limits, we find the average of *lower_lim* and *upper_lim* vectors as follows:

```
> ave_limits <- c(mean(lower_lim),mean(upper_lim))
> ave_limits
[1] 38.09635 43.74359
```

The average confidence interval limits of 1000 simulated intervals are very close to the limits computed from percentiles of empirical sampling distribution.

Simulating Confidence Intervals for population variance (σ^2)

Even though inferences concerning population variance (σ^2) and standard deviation (σ) are usually of less interest than that about the mean (μ), there are occasions where such inferences are needed. In this section we compute confidence intervals for variance and standard deviation using repeated sampling from our real data set and study the concepts. Bonett [2] has noted that the exact confidence intervals for σ^2 and σ given in many text books are sensitive to violations of the normality assumption and their performance does not improve with increasing sample size. We observed in the previous section that the distribution of *hrs2* data does not follow a normal distribution. This data has a very high peak in middle of the distribution. Use of this data allows us to study the robustness of the confidence intervals. Now we introduce the confidence intervals for σ^2 and σ.

First, we introduce the sampling distribution of sample variance S^2. When the original population is normally distributed, the random variable $\frac{(n-1)S^2}{\sigma^2} = \frac{\sum(X-\bar{X})^2}{\sigma^2}$ has a chi-square (χ^2) probability distribution with (n-1) degrees of freedom, Devore [5]. We will use our *hrs2* variable to compute the approximate sampling distribution of $\frac{(n-1)S^2}{\sigma^2}$ by taking 10,000 random samples. In this case we use 10,000 samples to examine the properties of the empirical sampling distribution more closely comparing to the theoretical distribution. The following R code selects the random samples of size 100, and draws the histogram.

```
> var_v = NULL
> for(i in 1:10000)
  {
+    x <- sample(hrs2, 100, replace = FALSE)
+    v <- 99*var(x)/var(hrs2)
+    var_v <- c(var_v,v)
  }
> hist(var_v, breaks = 30, main = expression(paste("Histogram of (n-
1)","S"^2,"/",sigma^2)), xlab = expression(paste("(n-1)","S"^2,"/",sigma^2)))
```

The theoretical sampling distribution for samples of size 100 is a chi-square distribution with 99 degrees of freedom. We will compare the approximate sampling distribution we obtained using the above R code with the theoretical distribution in *Figure 4*.

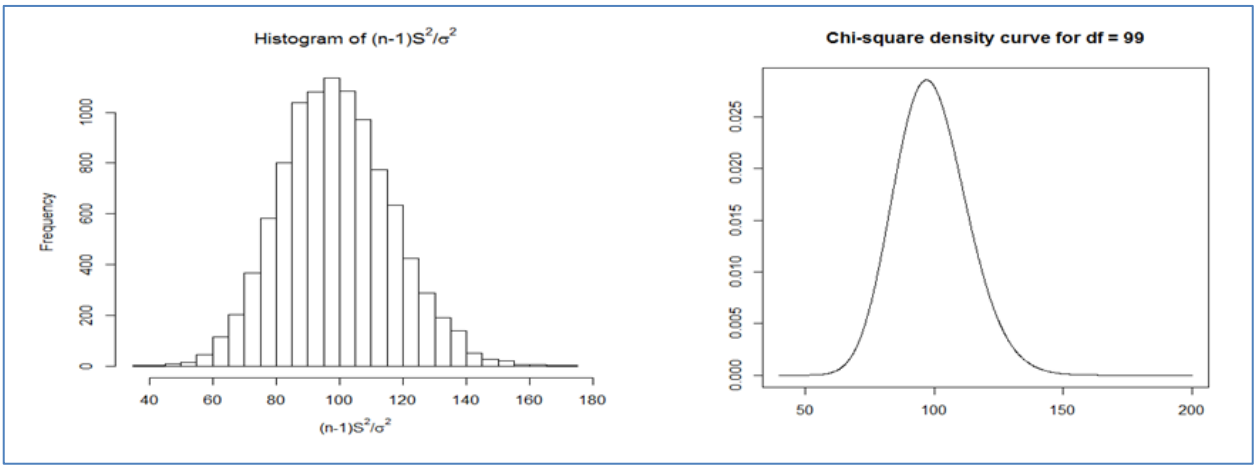

Figure 4: Empirical and Theoretical Sampling Distributions of $(n-1)S^2/\sigma^2$

From the above *Figure 4*, we observe that the empirical sampling distribution with samples of size 100, quite reasonably agree with the theoretical distribution even if the original population is not normally distributed.

Using the sampling distribution of $\frac{(n-1)S^2}{\sigma^2}$ as χ^2 with (n-1) degrees of freedom, the 100(1-α)% confidence interval for population variance σ^2 is given as $\left(\frac{(n-1)S^2}{\chi^2_{\alpha/2,n-1}}, \frac{(n-1)S^2}{\chi^2_{1-\alpha/2,n-1}} \right)$. The area under a chi-square curve with (*n-1*) degrees of freedom to the right of $\chi^2_{\alpha/2,n-1}$ is α/2, as is the area to the left of $\chi^2_{1-\alpha/2,n-1}$. Taking the square root of the endpoints of the above interval gives the 100(1-α)% confidence interval for σ. This confidence interval is valid when the original population has a normal distribution.

First we approximate the 95%confidence interval for variance σ^2 by identifying the 2.5th and 97.5th percentiles of the empirical distribution of S^2 for 10,000 samples. We have already calculated the empirical distribution of $\frac{(n-1)S^2}{\sigma^2}$. Multiplying this by $\sigma^2/(n-1)$ gives the distribution of S^2. The following R code computes the approximate confidence intervals for the variance and standard deviation:

```
> var_v1 <- var_v*var(hrs2)/99
> conflim <- c(quantile(var_v1,0.025), quantile(var_v1,0.975))
> conflim
   2.5%    97.5%
140.1664 283.0900
> sqrt(conflim)
   2.5%    97.5%
11.83919 16.82528
```

Based on the approximate confidence intervals, it can be concluded that we are 95% confident that the true population variance is between 140.17 hours and 283.09 hours and the standard deviation is between 11.94 hours and 16.83 hours. As we noted earlier, the inference procedures for σ^2 based on the assumption of a normally distributed population can work poorly if this assumption is violated. To investigate this further, we generate 10,000 random samples from population data and compute 95% confidence interval for each using the normal theory confidence interval formula. Then we compute the proportion of confidence intervals containing the true population variance. This coverage probability should be exact or close to 95% if the assumptions are met. The following R code performs this task and computes the coverage probability:

```
> lower_lim <- NULL
> upper_lim <- NULL
> for(i in 1:10000)
  {
+   x <- sample(hrs2,100,replace = FALSE)
+   l_lim <- 99*var(x)/qchisq(0.975,99)
+   u_lim <- 99*var(x)/qchisq(0.025,99)
+  lower_lim <- c(lower_lim,l_lim)
+  upper_lim <- c(upper_lim,u_lim)
  }
> count <- 0
> for(i in 1:10000){
+ if (lower_lim[i] < var(hrs2) & upper_lim[i] > var(hrs2))
+ count <- count + 1}
> count/10000
[1] 0.8877
```

This cord segment is similar to what we used in the previous section for computing coverage probability for the mean except that we use formulas to compute confidence intervals for variance. This indicates that in long run, only 88.77% of the 95% confidence intervals will contain the actual population variance. This computation confirms that confidence intervals for population variance and standard deviation are sensitive to violation of normality assumption. To observe the effect of sample size on this coverage probability, we take a range of sample sizes and compute the coverage probabilities. *Table 1* shows our findings:

Table 1: Coverage Probabilities

n	Coverage Probability
25	0.8735
50	0.8785
100	0.8877
200	0.8920
300	0.9086
400	0.9258
500	0.9364

From *Table 1*, we observe that the coverage probability increases as sample size increases. For sample sizes as large as 300, confidence intervals do not perform well. For a sample size of 500, coverage probability is close to the assumed confidence interval of 95%. On the other hand, the confidence interval for mean is less sensitive to a violation of normality. Since in real life the distributions of the data are unknown or do not follow normal distribution, for most cases, we need very large sample sizes for estimations of population variance and standard deviation.

Conclusions

In this paper we show how to use large sets of real data and R to teach the confidence intervals for mean and variance. Many students have difficulties grasping these concepts in introductory statistics classes. Students feel comfortable using real data in their lessons rather than fake or simulated data. R enables students to write simple codes and create visuals for understanding concepts. Students understand the concepts of confidence intervals by repeating the sampling process and computing coverage probabilities. Using this approach, students observe that while confidence intervals for mean are less sensitive to non-normality, confidence intervals for variance and standard deviation are more sensitive.

References

[1] Ang, K. C. (2010). Teaching and Learning Mathematical Modeling with Technology, *Proceedings of the 15th Asian Technology Conference in Mathematics,* Kuala Lumpur, Malaysia, 19-29.

[2] Bonett, D. G. (2006). Approximate Confidence Interval for Standard Deviation of Nonnormal Distributions, Computational Statistics and Data Analysis, 50, 775-782.

[3] Chandrakantha, Leslie. (2018), Simulating Sampling Distribution of the Mean in R. *The Electronic Journal of Mathematics and Technology* (EJMT), 12(2): p 309-321

[4] Connor, D., & Davies, N. (2002). An International Resource for Learning and Teaching, *Teaching Statistics*, 24(2), 59-61.

[5] Devore, J. L. Probability and Statistics for Engineering and Sciences, 9th Edition, Boston, MA: Cengage Learning, 2016.

[6] Fidler, F. (2006). Should Psychology Abandon *p* value and Teach CIs instead? Evidence Based Reforms in Statistics Education, *Proceedings of the Seventh International Conference in Teaching Statistics.* Voorburg, The Netherlands.

[7] Franklin, C. A., & Garfield, J. (2006). Developing Statistics Education Guidelines for pre K-12 and College Courses, *Thinking and Reasoning about Data and Chance: Sixty-eight NCTM yearbook,* 345-375, Reston, VA.

[8] GAISE (2005). Guidelines for Assessment and Instruction in Statistics Education Report. American Statistical Association, Alexandria, VA. http://www.amstat.org/education/gaise/

[9] Garfield, J., & Ben-Zvi, D. (2009). Helping Students Develop Statistical Reasoning: Implementing a Statistical Reasoning Learning Environment, *Teaching Statistics,* 31(3), 72-77.

[10] Garfield, J., delMas, R., & Chance, B. (1999). Tools for Teaching and Assessing Statistical Inferences. Retrieved from www.tc.umn.edu/~delma001/stat_tools/

[11] General Social Survey (GSS): http://gss.norc.org/

[12] Hallgren, K. A., (2103). Conducting Simulation Studies in the R Programming Environment. *Tutorial in Quantitative Methods for Psychology*, 9(2), 43-60.

[13] Mills, J. D. (2002). Using Computer Simulation Methods to Teach Statistics: A Review of the Literature. *Journal of Statistics Education (Online),* 10 (1). http://www.amstat.org/publications/jse/v10n1/mills.html

[14] Pfannkuch, M. (2011). The Role of Context in Developing Informal Statistical Inferential Reasoning: A Classroom Study, *Mathematical Thinking and Learning,* 13(1&2), 27-46.

[15] Scheaffer, R.L. (2001). Statistics Education; Perusing the past, embrasing the present, and charting the future, *Newsletter for the Section on Statistical Educati*on, 7(1). http://www.amstat.org/sections/educ/newsletter/v7n1/Persuing.html.

[16] Willett, J. B., & Singer, J. D. (1992). Providing a Statistical Model: Teaching Applied Statistics Using Real-world Data, *Statistics for the Twenty First Century,* 83-98. Washington DC, Mathematical Association of America.