COMPSCIX 415.2 Homework 9/Final

Rajat Jain

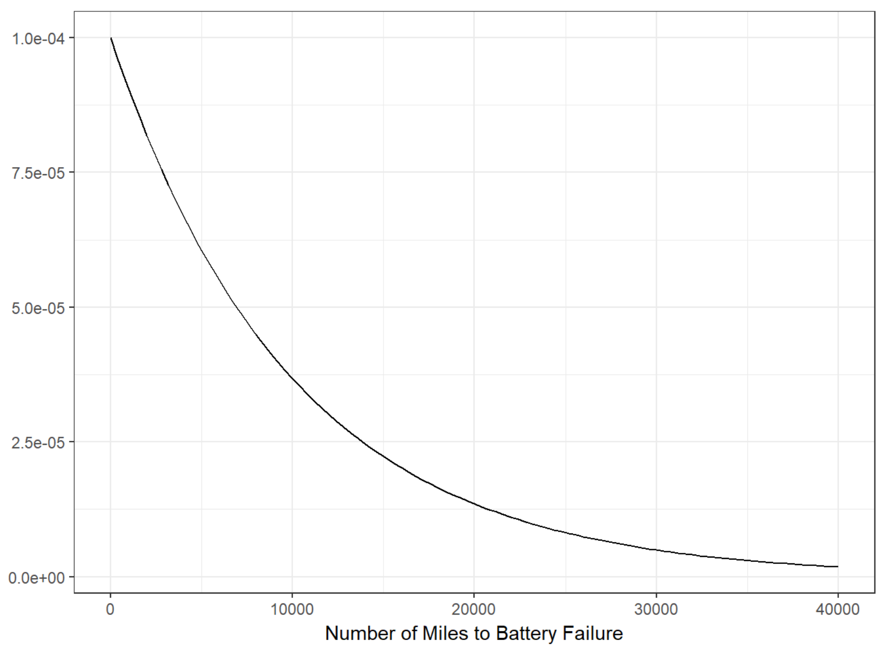
Aug 4th, 2018

Contents

Exercise 1 - Sampling Distributions, Functions and For Loops (10 points)

Recall that the distribution of the sample mean is approximately a Normal distribution, and that the standard error is $\frac{\sigma}{\sqrt{n}}$. This holds true regardless of the distribution of our population.

For this problem, assume that the number of miles that a particular car can run before its battery wears out is exponentially distributed with an average of 10,000 miles. The exponential distribution looks like this:



The exponential distribution has a rate parameter that controls how quickly the distribution decays and defines what the mean and standard deviation will be. In our case the `rate = 1/10000`, the `mean = 10000` and the `standard deviation = 10000`. You can sample from this exponential distribution in R using this code:

```
# sample size
samp_size <- 100
# set the rate parameter
samp_rate <- 1/10000

# take sample
rexp(n = samp_size, rate = samp_rate)
```

STEP 1

Write an R function that does the following:

- **Takes a sample of size `samp_size` from this exponential distribution (`samp_size` is an input parameter for the function)**
- **Calculates the mean of that sample**
- **Calculates the standard deviation of that sample**
- **Returns the calculated mean and standard deviation as a list**

Helper code

```
samp_fun <- function(samp_size, samp_rate) {

...your code here...

stats <- list(samp_avg = samp_avg, samp_std_dev = samp_std_dev)
return(stats)
}
```

Here is the code for the desired function:

```
samp_stats <- function(samp_size, samp_rate) {

  sample <- rexp(n = samp_size, rate = samp_rate)

  samp_avg <- mean(sample)
  samp_sd <- sd(sample)

  stats <- list(samp_avg = samp_avg, samp_std_dev = samp_sd)
  return(stats)
}
```

STEP 2

Then write a loop that does this:

- **Runs the above function 1000 times, with `samp_size = 50` and `samp_rate = 1/10000`**
- **Saves all of the sample means in a vector called `sample_means`, and all of the sample standard deviations in a vector called `sample_sds`**

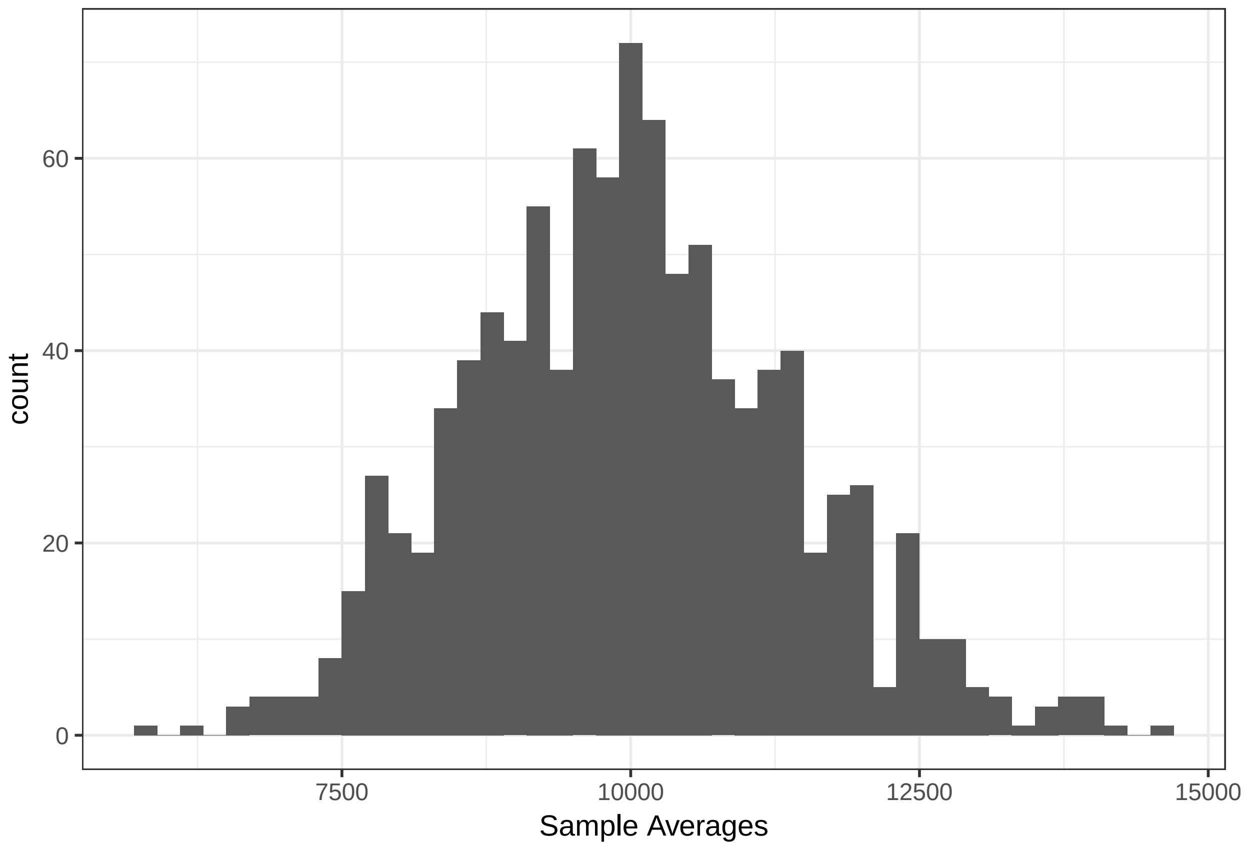
```
# No. of iterations.
N <- 1000
# Sample size.
n <- 50

# Output Vectors.
sample_means <- rep(NA, N)
sample_sds <- rep(NA, N)

for (i in 1:N) {
  stats <- samp_stats(n, 1/10000)
  sample_means[i] <- stats$samp_avg
  sample_sds[i] <- stats$samp_std_dev
}
```

STEP 3

Then

- **plot your sample means as a histogram**

```
ggplot(data.frame(samp_avg = sample_means), aes(x = samp_avg)) +
  geom_histogram(binwidth = 200) + xlab("Sample Averages") + theme_bw()
```



- **output the standard deviation of your sample means**

```
paste("Standard Deviation of sample means is", round(sd(sample_means), 2))
```

```
## [1] "Standard Deviation of sample means is 1398.41"
```

- **calculate the theoretical standard error** ($\sigma = 10000$, `n = sample size`)

```
paste("Theoretical Standard Error is", round(10000 / sqrt(n), 2))
```

```
## [1] "Theoretical Standard Error is 1414.21"
```

- **calculate the mean of the sample standard deviations and use this to calculate the empirical standard error**

```
paste("Mean of sample Standard Deviations is", round(mean(sample_sds), 2),
      "and Emperical Standard Error is", round(mean(sample_sds) / sqrt(n), 2))
```

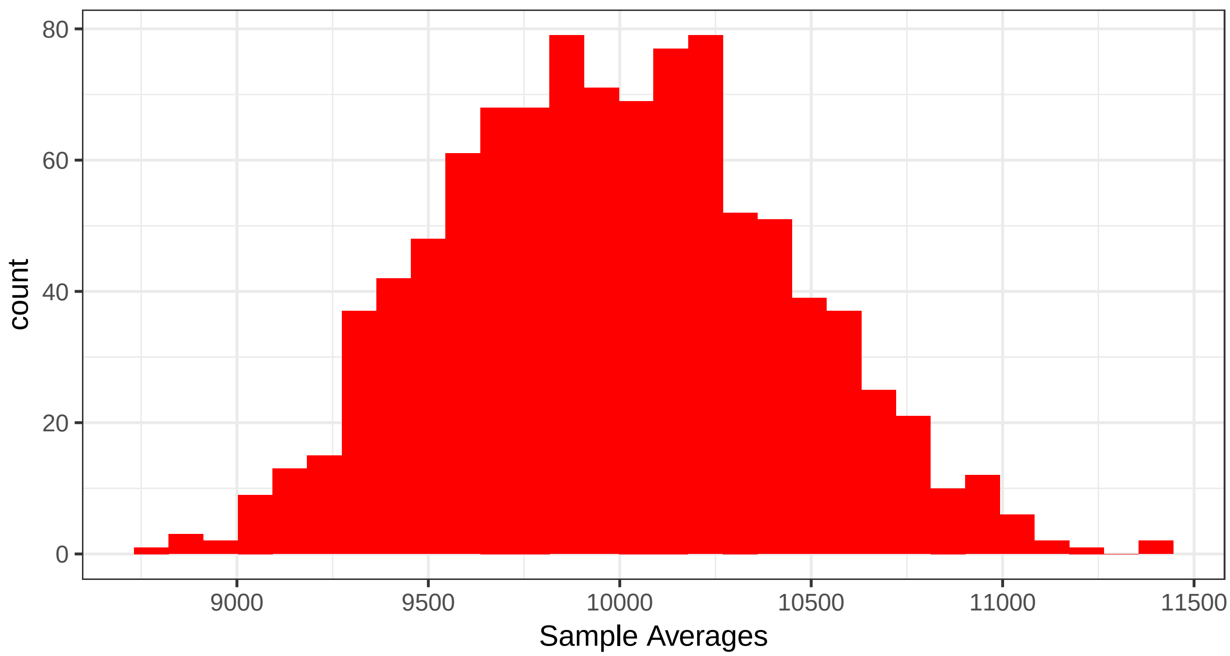
```
## [1] "Mean of sample Standard Deviations is 9849.74 and Emperical Standard Error is 1392.96"
```

STEP 4

Repeat `STEP 2` and `STEP 3` using a sample size of `500` and `5000`.

Repeating `STEP 2` and `STEP 3` using a sample size of 500:

```
n <- 500 # Sample size.
sample_means <- sample_sds <- rep(NA, N) # Output Vectors

# Run the function 1000 times, with samp_size = 500 and samp_rate = 1/10000
for (i in 1:N) {
  stats <- samp_stats(n, 1/10000)
  sample_means[i] <- stats$samp_avg
  sample_sds[i] <- stats$samp_std_dev
}

# Plot the sample means as a histogram
ggplot(data.frame(samp_avg = sample_means), aes(x = samp_avg)) +
  geom_histogram(fill = "red") + xlab("Sample Averages") + theme_bw()
```




```
# Output the standard deviation of the sample means
paste("Standard Deviation of sample means is", round(sd(sample_means), 2))
```

```
## [1] "Standard Deviation of sample means is 443.41"
```

```
# Calculate the theoretical standard error
paste("Theoretical Standard Error is", round(10000 / sqrt(n), 2))
```

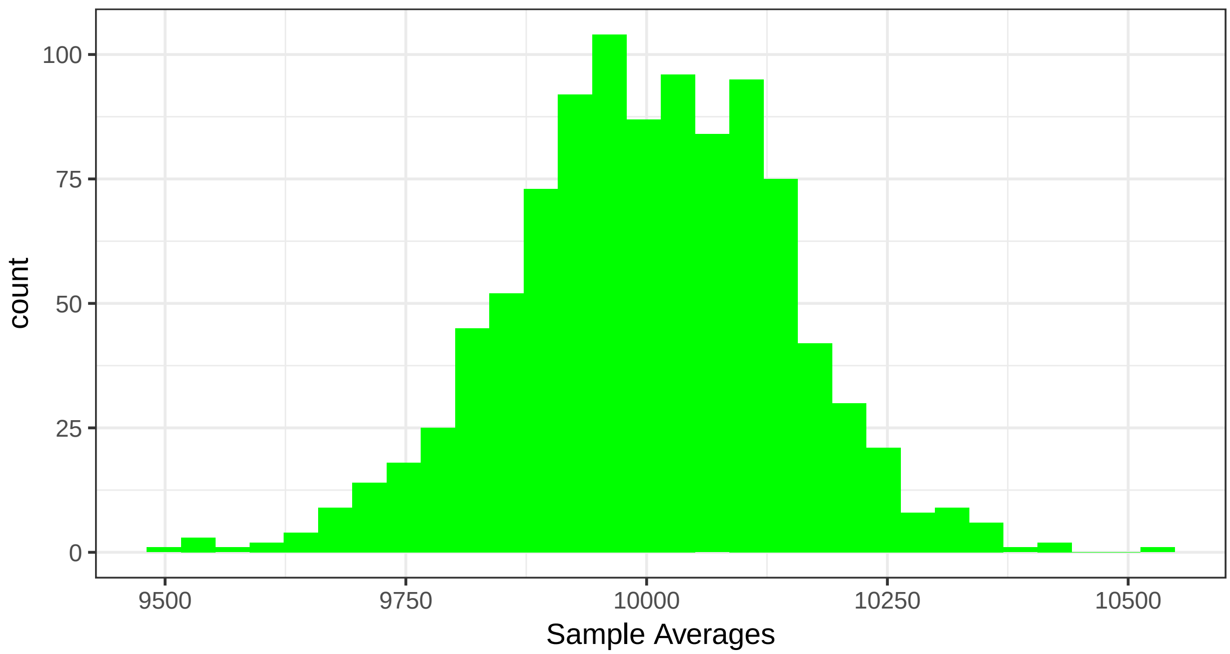
```
## [1] "Theoretical Standard Error is 447.21"
```

```
# Calculate the empirical standard error
paste("Mean of sample Standard Deviations is", round(mean(sample_sds), 2),
      "and Emperical Standard Error is", round(mean(sample_sds) / sqrt(n), 2))
```

```
## [1] "Mean of sample Standard Deviations is 9971.02 and Emperical Standard Error is 445.92"
```

Repeating `STEP 2` and `STEP 3` using a sample size of 5000:

```
n <- 5000 # Sample size.
sample_means <- sample_sds <- rep(NA, N) # Output Vectors

# Run the function 1000 times, with samp_size = 5000 and samp_rate = 1/10000
for (i in 1:N) {
  stats <- samp_stats(n, 1/10000)
  sample_means[i] <- stats$samp_avg
  sample_sds[i] <- stats$samp_std_dev
}

# Plot the sample means as a histogram
ggplot(data.frame(samp_avg = sample_means), aes(x = samp_avg)) +
  geom_histogram(fill = "green") + xlab("Sample Averages") + theme_bw()
```




```
# Output the standard deviation of the sample means
paste("Standard Deviation of sample means is", round(sd(sample_means), 2))
```

```
## [1] "Standard Deviation of sample means is 142.46"
```

```
# Calculate the theoretical standard error
paste("Theoretical Standard Error is", round(10000 / sqrt(n), 2))
```

```
## [1] "Theoretical Standard Error is 141.42"
```

```
# Calculate the empirical standard error
paste("Mean of sample Standard Deviations is", round(mean(sample_sds), 2),
      "and Emperical Standard Error is", round(mean(sample_sds) / sqrt(n), 2))
```

```
## [1] "Mean of sample Standard Deviations is 9997.92 and Emperical Standard Error is 141.39"
```

Exercise 2 - Linear Regression (5 points)

For this exercise we will return to the House Prices prediction dataset that we used for HW 7. You should have already downloaded the `train.csv` dataset before, but if you didn't you can download it from Canvas in this week's module.

Load the `train.csv` dataset into R and fit a regression model with:

- `y = SalePrice`
- **Features: `LotArea`, `OverallQual`, and `ExterQual`**

Here is the code to load data and fit a linear regression model:

```
train_data <- read_csv('train.csv')
model <- lm(formula = SalePrice ~ LotArea + OverallQual + ExterQual, data = train_data)
```

Answer these questions:

- **Use the broom package to output the coefficients and the R-squared**

```
tidy(model)
```

```
## # A tibble: 6 x 5
##   term        estimate std.error statistic   p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  40764.   12358.        3.30 9.95e-  4
## 2 LotArea          1.45     0.116     12.5  3.72e- 34
## 3 OverallQual  34466.    1216.        28.3  4.77e-141
## 4 ExterQualFa -95352.   14592.        -6.53 8.80e- 11
## 5 ExterQualGd -71529.    6737.       -10.6  2.05e- 25
## 6 ExterQualTA -97527.    7541.       -12.9  2.72e- 36
```

```
glance(model)
```

```
## # A tibble: 1 x 11
##   r.squared adj.r.squared  sigma statistic p.value    df  logLik    AIC
## *     <dbl>         <dbl>  <dbl>     <dbl>   <dbl> <int>   <dbl>  <dbl>
## 1     0.695         0.694 43948.      663.      0.     6 -17677. 35368.
## # ... with 3 more variables: BIC <dbl>, deviance <dbl>, df.residual <int>
```

- **Interpret the coefficient on `LotArea`**

Controlling for all other features, if the `LotArea` increases by 1 unit, the `SalePrice` increases by $1.45 on average.

- **Interpret the coefficient on `ExterQualGd`**

`SalePrice` for houses with `ExterQualGd`(Good exterior material quality) are on average $71,529.49 lower relative to the houses with `ExterQualEx`(Excellent) while controlling for all other features. *PS: There are no entries in training data with `ExterQual` value `Po`(Poor).*

- **Compare this model to the model we fit in HW 7 with `GrLivArea`, `OverallQual`, `Neighborhood`. Which is the better fitting model?**

This model has the adjusted-R-squared value of 0.69 which is lower than the adjusted-R-squared value of 0.78 achieved with the model we fit in HW 7 with `GrLivArea`, `OverallQual`, `Neighborhood`. It means that the current model explains less variability in the data as compared to the one built in HW 7. Hence, the model from HW 7 was a better fitting model.

Exercise 3 - AB Testing (5 points)

Download the `ab_test_data.csv` file from Canvas. This file contains two columns: `version` and `conversion`. Each row is a visitor to a webpage. The `version` column tells us which version of the webpage the visitor saw, and the `conversion` column is a binary value and equals `1` if the visitor converted (`0` otherwise).

We want to perform an AB test on this data to see if the conversion rates are different for the two versions of the webpage.

Reading the data file:

```r
ab_test_data <- read_csv('ab_test_data.csv')
```

```
## Parsed with column specification:
## cols(
##   version = col_character(),
##   conversion = col_integer()
## )
```

Answer these questions:

a. **What proportion of visitors converted for each version of the webpage?**

```r
ab_test_data %>%
  group_by(version) %>%
  summarise(
    conversion_rate = 100 * mean(conversion),
    num_converted   = sum(conversion),
    num_visited     = n()
    )
```

```
## # A tibble: 2 x 4
##   version conversion_rate num_converted num_visited
##   <chr>             <dbl>         <int>       <int>
## 1 A                  4.15            83        2000
## 2 B                 10.0            200        2000
```

4.15% visitors converted on version A and 10% visitors converted on version B of the webpage.

b. **Perform the AB test in R. What is the p-value for the AB test (hypothesis test of proportions)?**

Performing the A/B test using proportion test:

```r
test_results <- prop.test(c(83, 200), c(2000, 2000))

# p-value
test_results$p.value
```

```
## [1] 8.479709e-13
```

Since the p-value for the A/B test is less that 0.05 (α), we can reject the Null Hypothesis that version A and B of the webpage have same conversion rate. This means that our alternate hypothesis is accepted and version A and B have statistically significant differences in conversion rates.