HUDM6026 Homework_01

Chenguang Pan

Jan 23, 2023

Question 01

A random variable X has a chi-square distribution with k degrees of freedom (df) if it can be expressed as the sum of k independent squared random normal variables. In that case, we write that X~x2k. Write a function that has two arguments, n for sample size and k for degrees of freedom. The purpose of the function is to generate a random sample of size n from a x2k distribution. Importantly, you may not use the rchisq suite of functions. Instead, use ***rnorm()*** *and the definition given above.*

MY SOLUTION:
Based on the chi-square distribution's definition, we randomly draw n variables from a standard normal distribution for k times, and adds the square sum of them, then the distribution of this sum will follow a chi-square distribution. Based on this logic, I write a function containing a for k-loop to simulate this process.

```
> my_chisqr <- function(n, k){
+   # create a vector with n 0s
+   Q <- rep(0,n)
+   # use a loop to k times sampling
+   for (df in c(1:k)){
+     # create a vector of RVs from a standard normal distribution
+     rv_vec <- rnorm(n, mean = 0, sd = 1)
+     # square the randomly RVs and adds them up within the for loop
+     Q <- Q + rv_vec^2
+   }
+   return(Q)
+ }
```

Question 02

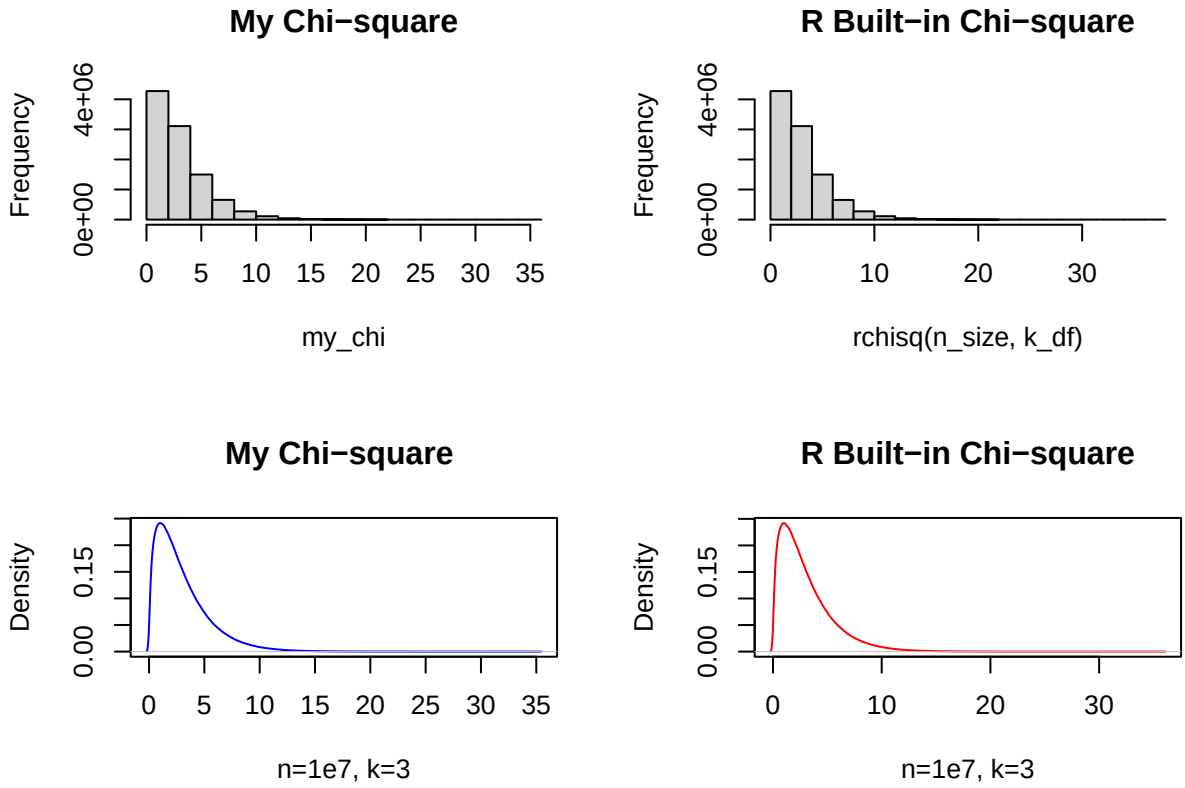
Use your function from above to generate a very large sample (e.g., n = 1e7) from x23. Display the distribution of the sample by creating a histogram in base or ggplot2. Play with the number of breaks until you settle on a number that looks good. Then, use rchisq() to generate a large sample of the same size and also create a histogram. If your function is working correctly, the distributions should be very nearly identical.

MY SOLUTION:
I will test the function with `n = 1e7` and `k = 3`. To ensure replication, using the random `set.seed(1000)` and compare it with built-in function `rchisq()`.

```
> set.seed(1000)
> n_size <- 1e7
> k_df <- 3
```

```
> # using the function above to get the chi-square with n size and k degree freedom
> my_chi <- my_chisqr(n_size, k_df)
> # plot the the results by using ggplot and plot(density) to make the figure looking good
> par(mfrow = c(2,2))
> # draw the histogram
> hist(my_chi, breaks=20, main = "My Chi-square")
> hist(rchisq(n_size, k_df), breaks = 20, main = "R Built-in Chi-square")
> plot(density(my_chi), col = "blue",
+      main = "My Chi-square ",
+      xlab = 'n=1e7, k=3',
+      ylab = 'Density')
> plot(density(rchisq(n_size, k_df)), col = "red",
+      main = "R Built-in Chi-square",
+      xlab = 'n=1e7, k=3',
+      ylab = 'Density')
> # to add a main title for all four graphs
> mtext("Figure 1. Graphs For Question 2",
+       side = 3,
+       line = -1,
+       outer = T)
```

Figure 1. Graphs For Question 2



These figures shows that my function works well and the results are identical to the R built-in function `rchisq()`.

Question 03

The chi-square distribution with 1 df is skewed heavily to the right. Generate 10000 sample means from iid samples of x21 of size n = 5 and plot the distribution of the sample means using a histogram or other univariate density plotting method. Then do the same for n = 10, n = 20, and n = 40. Superimpose a normal curve with mean equal to sample mean of the means and variance equal to sample variance of the means for each plot. Discuss whether this progression demonstrates the LLNs or the CLT and why.

MY SOLUTION:

First, I will write a funciton to draw sample for 10000 times from a chi-square distribution with the size of `size` and `k = 1`. Then, calculate the sample means. That is, this function will return the 10000 samples' means.

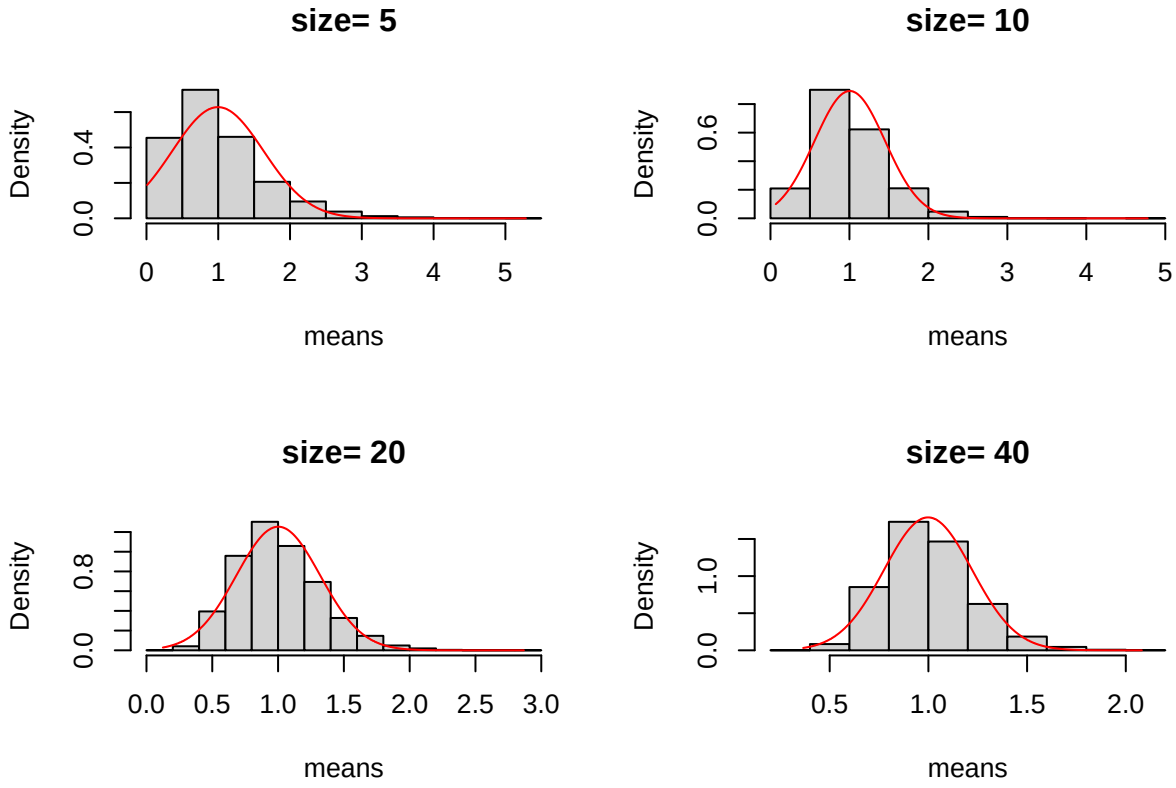
```
> # create a vector to load the 1e5 means
> means_set <- rep(0, 10000)
> # use a for loop to draw sample for 1e5 times
> get_means <- function(size){
+   for (i in c(1:10000)) {
+     # draw a sample from a chisqr distribution with size and 1 df
+     rv_vec <- rchisq(size, 1)
+     means_set[i] <- mean(rv_vec)
+   }
+   return(means_set)
+ }
```

This function `get_means()`runs well. Then, I use a for-loop to draw the four histograms.

From the following four histograms, one can see all the 10000 samples' means converge at the peak. But the distribution of the means is not followed a normal distribution. Therefore, these figures demonstrates that LLNs rather than CLT.

```
> par(mfrow = c(2,2))
> for (size_ in c(5, 10, 20, 40)) {
+   mean_vec <- get_means(size_)
+   # add argument prob = TRUE to draw density hist.
+   hist(mean_vec, prob = TRUE, breaks = 10,
+        main = paste("size=", as.character(size_)),
+        xlab = "means")
+   x <- seq(min(mean_vec),max(mean_vec), length = length(mean_vec))
+   f <- dnorm(x, mean = mean(mean_vec), sd = sd(mean_vec))
+   lines(x,f, col = 'red', lwd = 1)
+ }
> mtext("Figure 2. Histograms For Question 3",
+       side = 3,
+       line = -1,
+       outer = T)
```

Figure 2. Histograms For Question 3



Question 04 [bonous]

Write a function to generate data (in a data frame or tibble) from the following regression model.

MY SOLUTION:
This question seems to be more challenging. I will write all my thoughts behind my codes.

First things first, why we need to generate data based on the model's assumption? This website mentioned: one of the main problems that the researchers usually encountered when trying to implement the proposed model is the lack of the proper real-world dataset that follows the model's assumptions. Or in the other case, the real-world dataset exists, but the dataset itself is very expensive and hard to collect.

The process of generating a simulated dataset can be explained as follows. First, we specify the model that we want to simulate. Next, we determine each independent variable's coefficient, then simulate the independent variable and error that follows a probability distribution. And finally, compute the dependent variable based on the simulated independent variable (and its predetermined coefficient) and error.

Given that sovling this question need some knowledge about multivariate normal distribution. I found it here to refresh my understanding.

For the code, I made a function to generate data in matrix form.

```
> # import the package to run multivariate normal distribution
>
> data_gen <- function(n){
+   library(mvtnorm)
+   # define the covariance matrix of X1, X2, and X3
```

```
+   # note X1~X3 are all continuous variables.
+   cov_matrix <- matrix(c(13.6, -14.5, -103.4,
+                          -14.5, 65.2, 154.0,
+                          -103.4, 154.0, 2702.0),3,3)
+   # generate the X's matrix using multivariate normal distribution
+   X <- rmvnorm(n = n, sigma = cov_matrix) # TODO: what about the means?
+   # X is a n * 3 matrix
+   X1 <- X[,1]; X2 <- X[,2]; X3 <- X[,3]
+   # generate the error term
+   Error <- rnorm(n = n, mean = 0, sd = 0.75)
+   # estimate the vector of dependent variable based on the model
+   Y <- 71.0 - 0.28*X1 + 0.05*X2 - 0.007*X3 + Error
+   out <- data.frame(X1,X2,X3,Y)
+   return(out)
+ }
```

This function runs well. Then, create a dataset with 1000 observations and test the linear model.

```
> set.seed(666)
> data_simulate <- data_gen(1000)
> # head(data_simulate)
> lm_est <- lm(Y ~ X1 + X2 + X3, data = data_simulate)
> summary(lm_est)

Call:
lm(formula = Y ~ X1 + X2 + X3, data = data_simulate)

Residuals:
     Min       1Q   Median       3Q      Max
-2.13684 -0.47060 -0.02534  0.50065  2.35442

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 70.9846997  0.0238692 2973.90   <2e-16 ***
X1          -0.2832359  0.0086811  -32.63   <2e-16 ***
X2           0.0514251  0.0034581   14.87   <2e-16 ***
X3          -0.0078670  0.0005578  -14.11   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7541 on 996 degrees of freedom
Multiple R-squared:  0.6807,    Adjusted R-squared:  0.6798
F-statistic: 707.8 on 3 and 996 DF,  p-value: < 2.2e-16
```

The results show that the estimated coefficient are $\beta_0 = 70.98$, $\beta_1 = -0.28$,$\beta_2 = 0.05$, $\beta_3 = 0.008$, which are pretty close to the coefficients the text given. The overall model can significantly predict the outcome. All independent variables and constant are statistically significant as well (based on the P-value at the significant level of .05).