SA1 Statistical Computing

Bayquen, Christopher Gilbert A. Billones, Cristel Kaye

2024-06-26

Use Monte Carlo Simulation to investigate whether the empirical Type I error rate of the t-test is approximately equal to the nominal significance level α, when the sampled population is non-normal. The t-test is robust to mild departures from normality. Discuss the simulation results for the cases where the sampled population is (i) $X^2(1)$, (ii) Uniform(0,2), and (iii) Exponential(rate=1). In each case, test $H_0 : \mu = \mu_0$ vs $H_1 : \mu \neq \mu_0$, where μ_0 is the mean of X^2 (1), Uniform(0,2), and Exponential(1), respectively.

```
# Different sample sizes
num <- c(25, 50, 100, 200, 500, 1000, 5000)
# Different number of Monte Carlo simulations
m_values <- c(100, 500, 1000, 10000)

# Set seed for reproducibility
set.seed(123)

# Initialize an empty list to store the error tables for
# different m values
err_tables <- list()

for (m in m_values) {
    results <- data.frame(Distribution = character(), Sample_Size = integer(),
        Type_I_Error_Rate = numeric(), Standard_Error = numeric(),
        stringsAsFactors = FALSE)

    for (n in num) {
        cv <- qt(0.975, n - 1)

        # Estimate the Type-I error for chi-square
        # distribution
        er1_vals <- sapply(1:m, FUN = function(o) {
            x <- rchisq(n, 1)
            m_x <- mean(x)
            sd <- sqrt(var(x))

            abs((m_x - 1) * sqrt(n)/sd) >= cv
        })
        er1 <- mean(er1_vals)
        se1 <- sd(er1_vals)/sqrt(m)

        # Estimate the Type-I error for uniform
        # distribution
        er2_vals <- sapply(1:m, FUN = function(o) {
            x <- runif(n, 0, 2)
```

```
            m_x <- mean(x)
            sd <- sqrt(var(x))

            abs((m_x - 1) * sqrt(n)/sd) >= cv
        })
        er2 <- mean(er2_vals)
        se2 <- sd(er2_vals)/sqrt(m)

        # Estimate the Type-I error for exponential
        # distribution
        er3_vals <- sapply(1:m, FUN = function(o) {
            x <- rexp(n, 1)
            m_x <- mean(x)
            sd <- sqrt(var(x))

            abs((m_x - 1) * sqrt(n)/sd) >= cv
        })
        er3 <- mean(er3_vals)
        se3 <- sd(er3_vals)/sqrt(m)

        # Store the results
        results <- rbind(results, data.frame(Distribution = "chi(1)",
            Sample_Size = n, Type_I_Error_Rate = er1, Standard_Error = se1),
            data.frame(Distribution = "U(0,2)", Sample_Size = n,
                Type_I_Error_Rate = er2, Standard_Error = se2),
            data.frame(Distribution = "exp(1)", Sample_Size = n,
                Type_I_Error_Rate = er3, Standard_Error = se3))
    }

    # Store the error table in the list with the number of
    # simulations as the name
    err_tables[[paste0("m_", m)]] <- results
}

# Access individual data frames
df_100 <- err_tables[["m_100"]]
df_500 <- err_tables[["m_500"]]
df_1000 <- err_tables[["m_1000"]]
df_10000 <- err_tables[["m_10000"]]
```

Code Documentation

Provided the code above, multiple Monte Carlo simulations were ran to generate random numerical results to investigate the empirical Type I error rate of the t-test when the sampled population is non-normal (chi-square, exponential, and uniform distribution); this Type I error rate is the probability of incorrectly rejecting the null hypothesis when it is true. For each of the non-normal distribution tested, The algorithm below is designed to calculate the empirical Type I error rate and its standard error for a t-test.

```
err_vals <- sapply(1:m, FUN = function(o) {
    x <- distribution_function
    m_x <- mean(x)
    sd <- sqrt(var(x))

    abs((m_x - 1) * sqrt(n)/sd) >= cv
})
err <- mean(err_vals)
se <- sd(err_vals)/sqrt(m)
```

Initially, a critical value cv is computed using the `qt(0.975, n-1)` function provided in R, for a 95% confidence level and an n-1 degree of freedom to determine the rejection region of the t-test. The entire process of generating the sample, computing the test statistic, and comparing it to the critical value is repeated `m` times using the sapply function.

In the algorithm shown above:

x <- distribution_function, generates a sample x of size n from a non-normal distribution. The function that follows will be deterined depending on which non-normal distribution is being tested.

- For a Chi-Square distribution with 1 degree of freedom: `x <- rchisq(n, 1)`
- For a Uniform distribution between 0 and 2: `x <- runif(n, 0, 2)`
- For an Exponential distribution with rate 1: `x <- rexp(n, 1)`

m_x <- mean(x), calculates the sample mean from the generated x.
sd <- sqrt(var(x)), calculates the sample standard deviation; the square root of the variance of x.
abs((m_x - 1) sqrt(n) / se) >= cv, computes the test statistic for the t-test as

$$\text{Test Statistic} = \left| \frac{(\bar{x} - \mu_0)\sqrt{n}}{s} \right|$$

- where $\bar{x}$ is the sample mean (m_x)
- μ_0 is the population mean under the null hypothesis; for the given distributions $\mu_0 = 1$
- n is the sample size
- s is the standard deviation (sd).

err_vals is a logical vector indicating whether the null hypothesis was rejected (`TRUE`) or not (`FALSE`) in each simulation.

The empirical Type I error rate, **err**, is the mean of the logical vector `err_vals`, representing the proportion of times the null hypothesis was incorrectly rejected.

se, The standard error of the empirical Type I error rate is calculated as the standard deviation of `err_vals` divided by the square root of `m`. This gives an estimate of the variability in the Type I error rate.

This algorithm is then used for each of the non-normal distributions inside the Monte Carlo simulation loop that we have initialized in the prior code. We have tested for each of the non-normal distributions using different sample sizes (`25, 50, 100, 200, 500, 1000, 5000`), and also tested for different iterations of the Monte Carlo simulation (`100, 500, 1000, 10000`). Each iteration of the Monte Carlo simulation is then stored into each of their own data frames to be analyzed.

Analysis and Visualization