Aadhil Ahmed

2024-04-06

Table of Contents

Import Libraries

```
## Warning: package 'tidyverse' was built under R version 4.4.1

## Warning: package 'ggplot2' was built under R version 4.4.1

## Warning: package 'tibble' was built under R version 4.4.1

## Warning: package 'tidyr' was built under R version 4.4.1

## Warning: package 'readr' was built under R version 4.4.1

## Warning: package 'purrr' was built under R version 4.4.1

## Warning: package 'dplyr' was built under R version 4.4.1

## Warning: package 'stringr' was built under R version 4.4.1
## Warning: package 'forcats' was built under R version 4.4.1

## Warning: package 'lubridate' was built under R version 4.4.1

## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.4     ✓ readr     2.1.5
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ ggplot2   3.5.1     ✓ tibble    3.2.1
## ✓ lubridate 1.9.3     ✓ tidyr     1.3.1
## ✓ purrr     1.0.2
## ── Conflicts ──────────────────────────────────────────
tidyverse_conflicts() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors

## Warning: package 'simmer' was built under R version 4.4.1

##
## Attaching package: 'simmer'
##
## The following objects are masked from 'package:lubridate':
##
##     now, rollback
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## The following object is masked from 'package:tidyr':
##
##     separate

## Warning: package 'simmer.plot' was built under R version 4.4.1

##
## Attaching package: 'simmer.plot'
##
## The following objects are masked from 'package:simmer':
##
##     get_mon_arrivals, get_mon_attributes, get_mon_resources

## Warning: package 'gridExtra' was built under R version 4.4.1

##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
```

```
##
##     combine
```

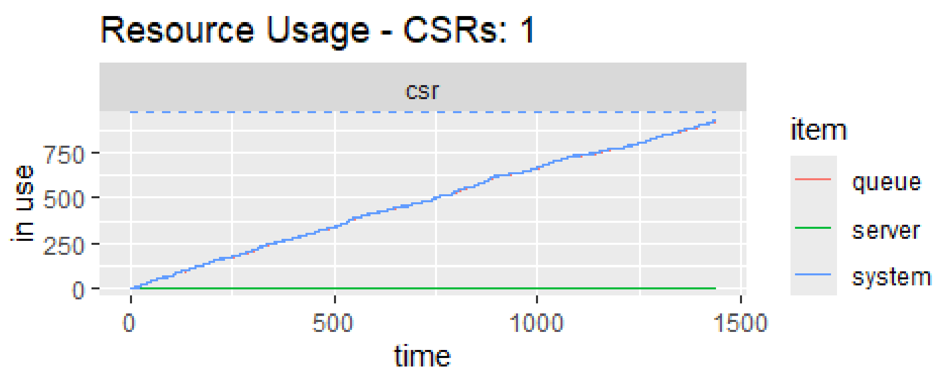
Question 1

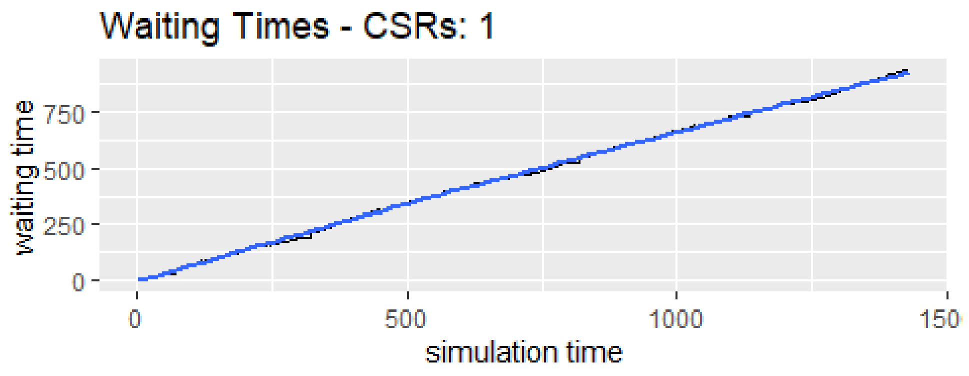
Task A) Simualtion of Call centre Customer Service

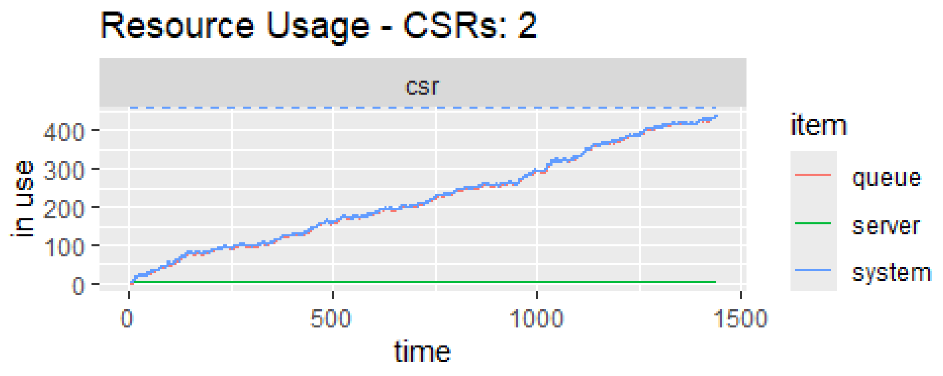
```
# Set seed
set.seed(123)

run_call_center_simulation <- function(num_CSRS) {
  # Initialize a simulation environment for the call centre
  env <- simmer("call centre")

  # Define the trajectory for a customer in the call centre simulation
  customer_arrival <- trajectory("customer") %>%
    seize("csr", 1) %>%  # Customer seizes one CSR
    timeout(function() rexp(1, rate = 1/3)) %>%
    # Service time = exponential distribution (mean service rate 1/3)
    release("csr", 1)  # Customer releases the CSR

  # Add CSRs
  env <- env %>%
    add_resource("csr", capacity = num_CSRS)

  # Add a generator for customers using the defined trajectory
  env <- env %>%
    add_generator("customer", customer_arrival, function() rpois(1, lambda =
1))
    # poisson arrival rate with mean of 1 customer per minute

  # Run the simulation for 1 days
  env %>%
    run(until = 1440)

  # Plot resource (CSR) usage over time
  plot_resource_usage <- plot(get_mon_resources(env), metric = "usage", step
= T) +
    ggtitle(paste("Resource Usage - CSRs:", num_CSRS))

  # Plot customer waiting times
  plot_waiting_times <- plot(get_mon_arrivals(env), metric = "waiting_time")
+
    ggtitle(paste("Waiting Times - CSRs:", num_CSRS))

  # Arrange the plots side by side in a single plot
  grid.arrange(plot_resource_usage, plot_waiting_times, ncol = 1)
}
```

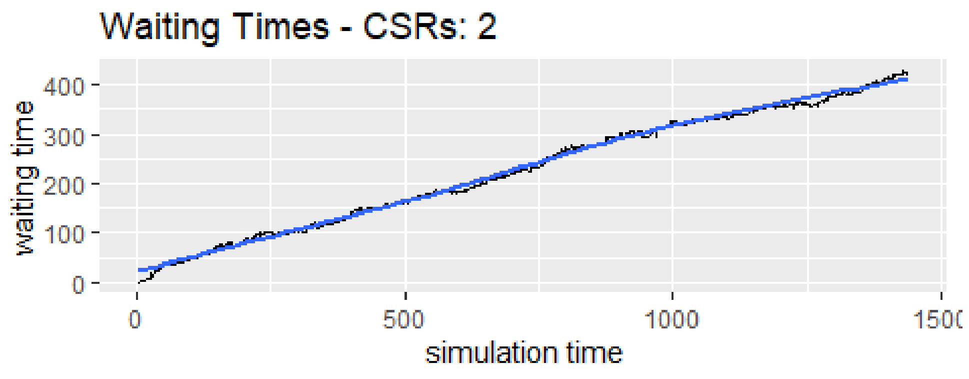
```r
for (num_CSRS in 1:10) {
  run_call_center_simulation(num_CSRS)
}
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

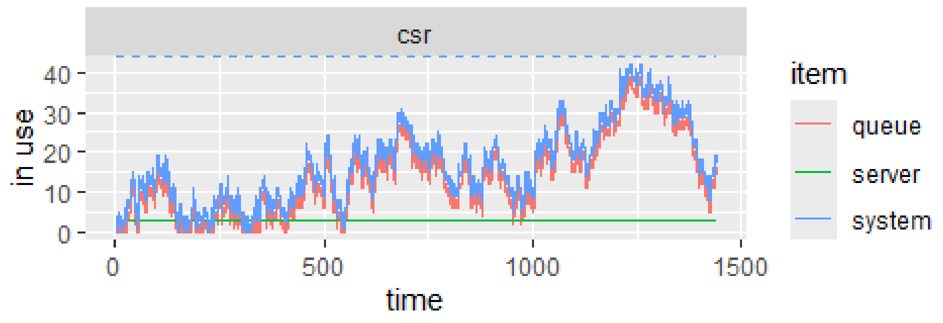

Resource Usage - CSRs: 1
csr
in use
750
500
250
0
0
500
1000
1500
time
item
queue
server
system


Waiting Times - CSRs: 1
waiting time
750
500
250
0
0
500
1000
150
simulation time

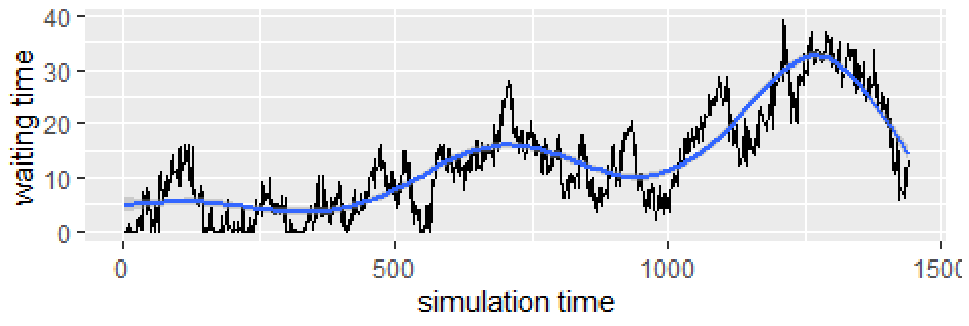

Resource Usage - CSRs: 2
csr
in use
400
300
200
100
0
0
500
1000
1500
time
item
queue
server
system


Waiting Times - CSRs: 2
waiting time
400
300
200
100
0
0
500
1000
150
simulation time

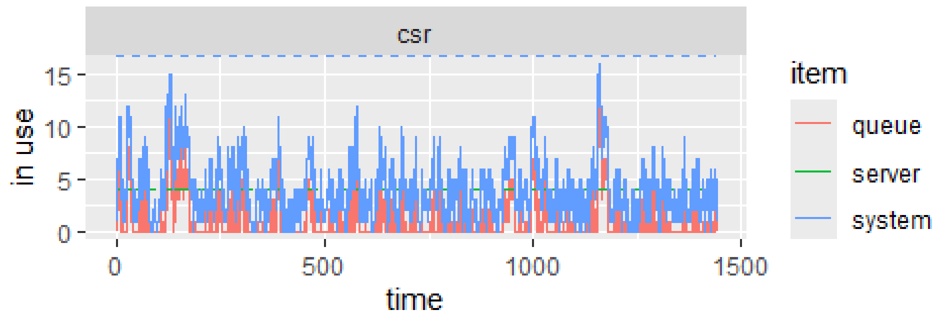
Resource Usage - CSRs: 3



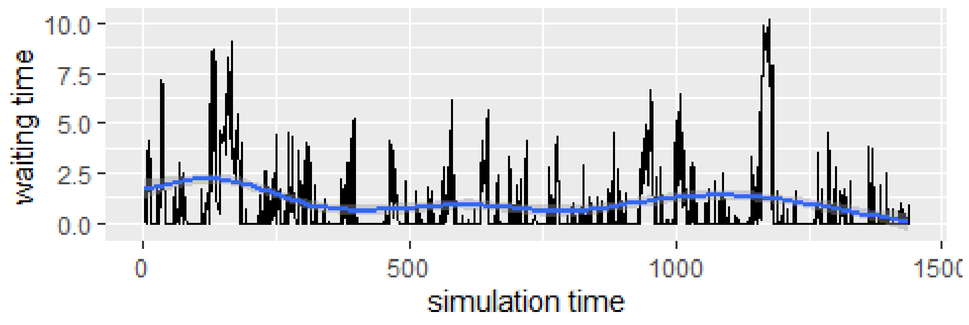
Waiting Times - CSRs: 3

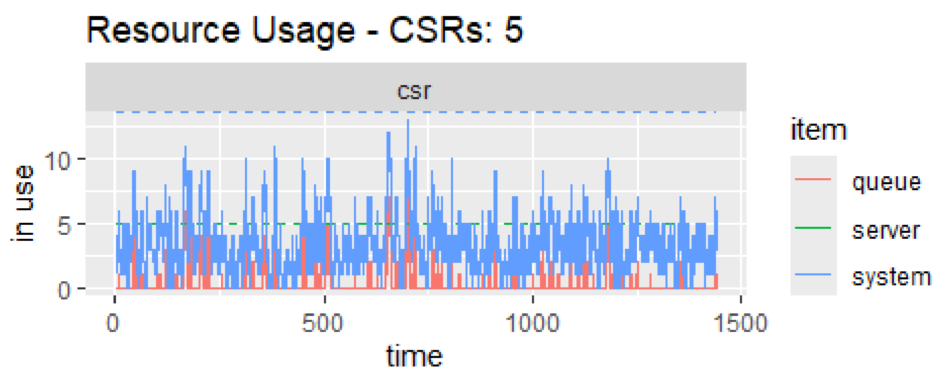


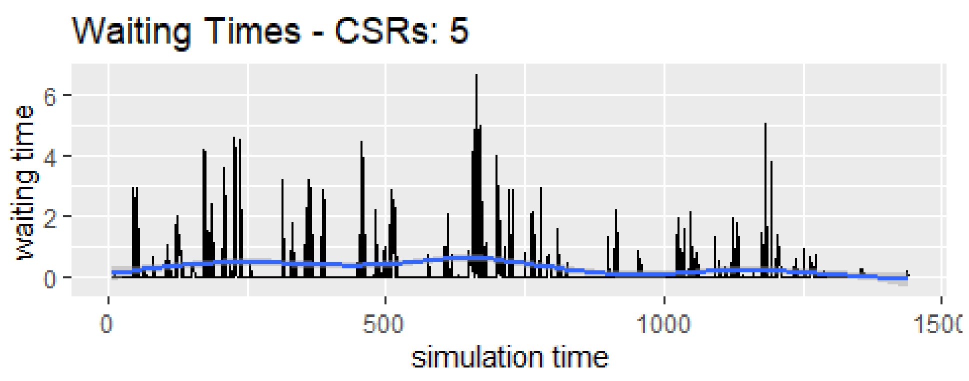
Resource Usage - CSRs: 4

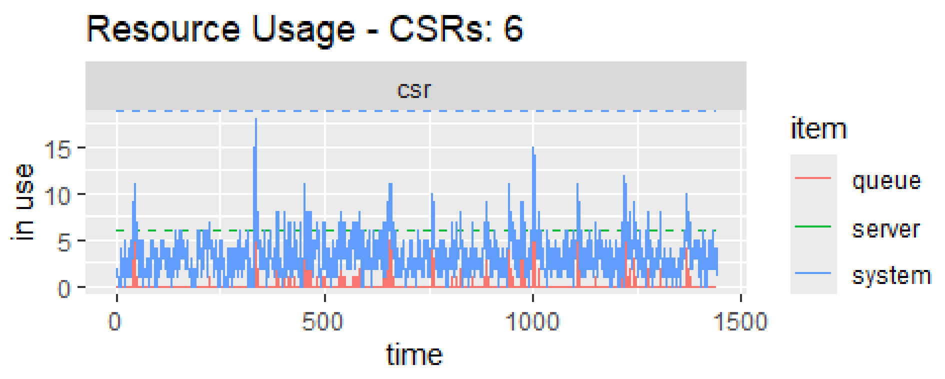


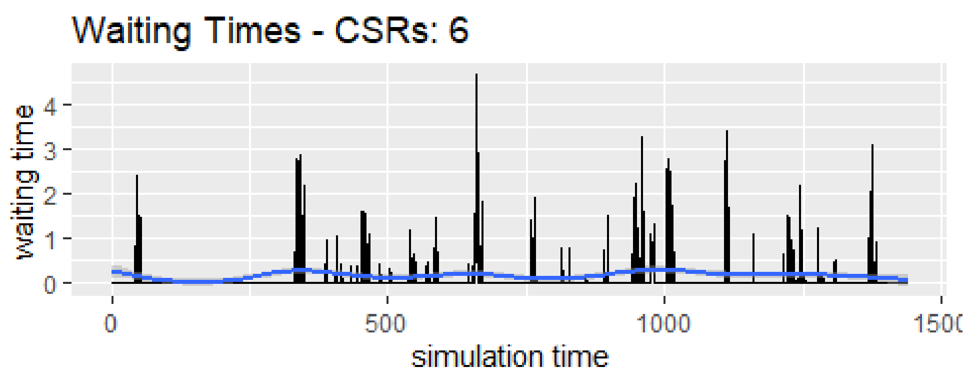
Waiting Times - CSRs: 4

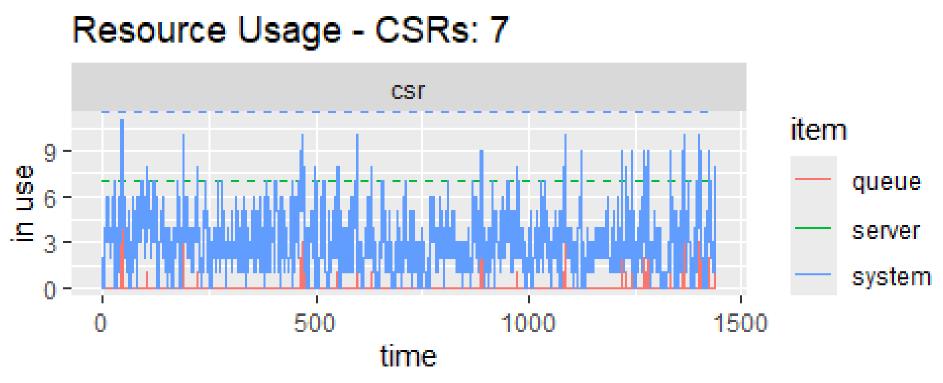


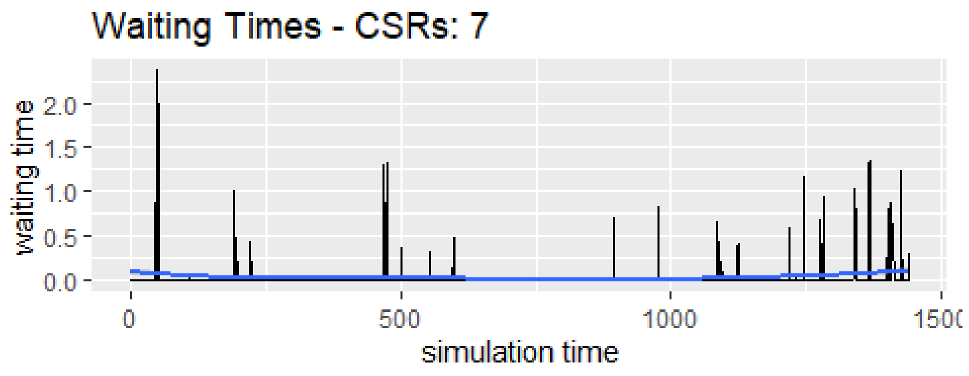

Resource Usage - CSRs: 5
csr
in use
10
5
0
0
500
1000
1500
time
item
queue
server
system

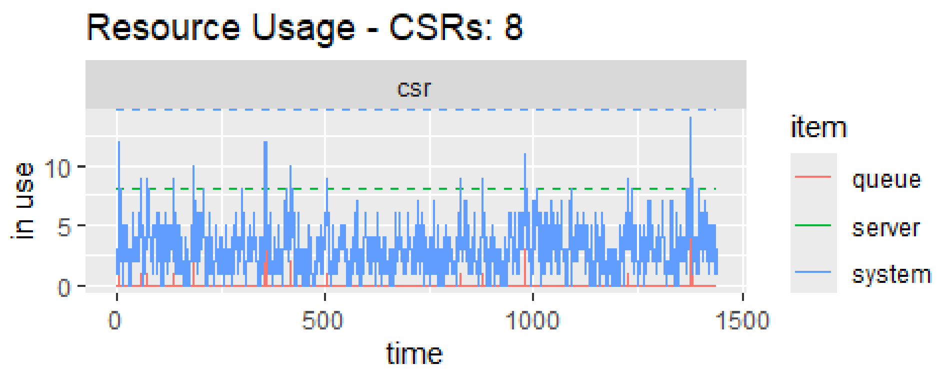

Waiting Times - CSRs: 5
waiting time
6
4
2
0
0
500
1000
1500
simulation time

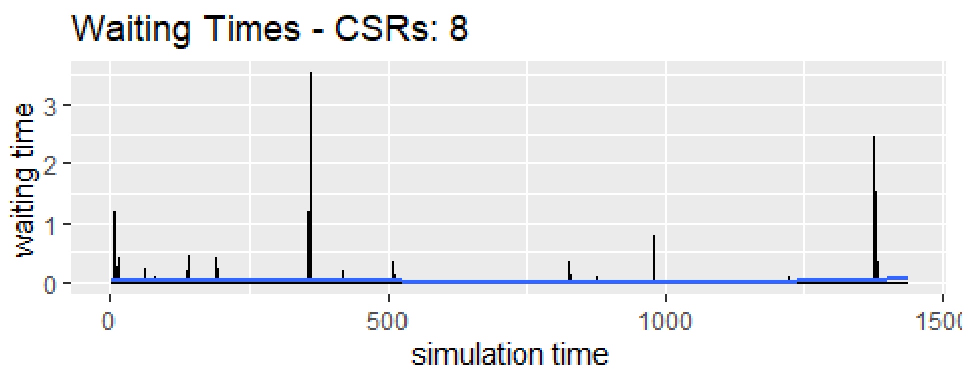

Resource Usage - CSRs: 6
csr
in use
15
10
5
0
0
500
1000
1500
time
item
queue
server
system

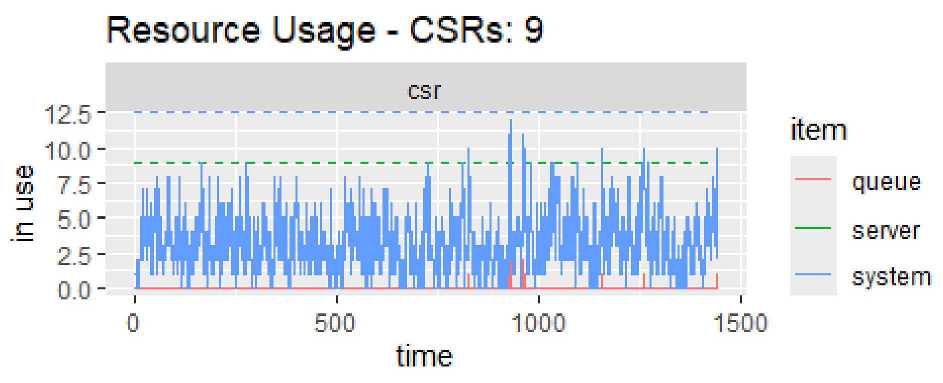

Waiting Times - CSRs: 6
waiting time
4
3
2
1
0
0
500
1000
1500
simulation time

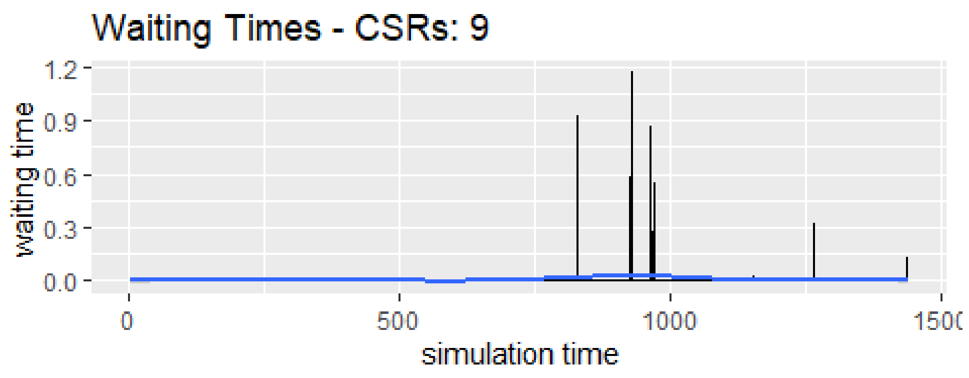

Resource Usage - CSRs: 7
csr
in use
9
6
3
0
0
500
1000
1500
time
item
queue
server
system

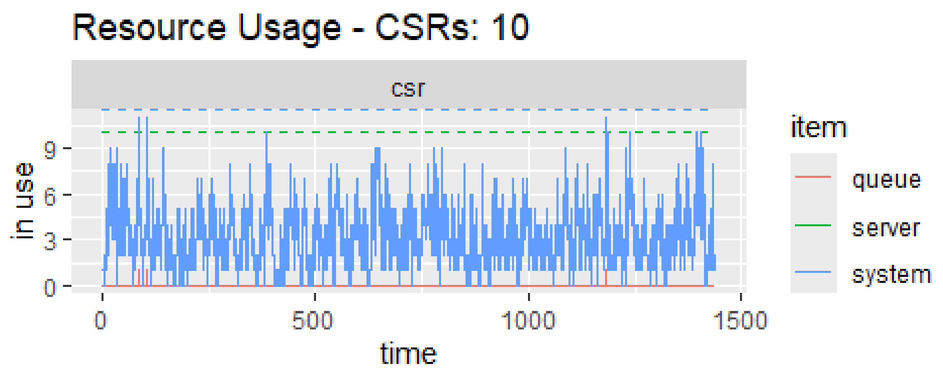

Waiting Times - CSRs: 7
waiting time
2.0
1.5
1.0
0.5
0.0
0
500
1000
150
simulation time

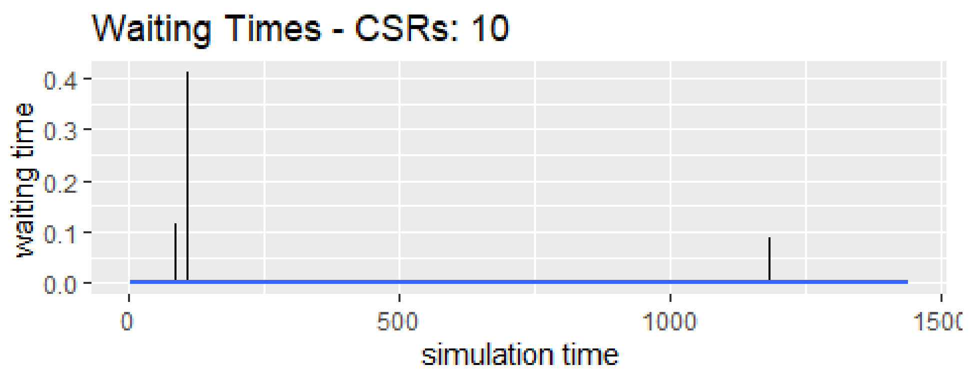

Resource Usage - CSRs: 8
csr
in use
10
5
0
0
500
1000
1500
time
item
queue
server
system


Waiting Times - CSRs: 8
waiting time
3
2
1
0
0
500
1000
150
simulation time









Task B) Discussion of results

From the simulated plot we can be sure that 1 - 2 CSRs yield little or no results resulting in providing customer service while having 3 - 5 CSRs although the waiting time is reduced and the customer service can be provided to certain extent employing 6+ CSRs reduces and eliminated waiting time for the customers its is also notable that plot 6 - 10 stays constant which indicates there is no value in hiring more CSRs as there is no noticable benifit in doing so hiring 6 CSRs seems to be the ideal solution to eliminating customer waiting time while saving companyy resources.

Question 2

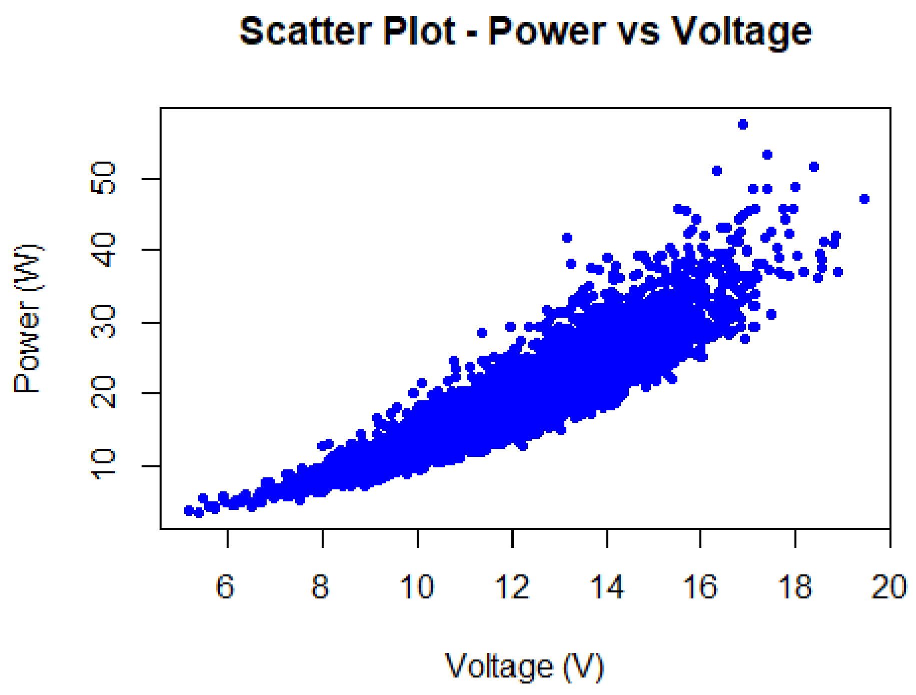
Task A) Power dissipated in a resistor (Generate 7000 values)

```
# Set seed
set.seed(123)

# Define parameters for voltage and resistance distributions
mean_voltage <- 12 # 12V
sd_voltage <- 2 # 2V
mean_resistance <- 8 # 8R
sd_resistance <- 1 # 1R

# Generate random samples for voltage and resistance
voltage <- rnorm(7000, mean = mean_voltage, sd = sd_voltage)
resistance <- rnorm(7000, mean = mean_resistance, sd = sd_resistance)

# Calculate power dissipated using the formula: power = (voltage^2) /
resistance
power <- (voltage^2) / resistance
print(power[1:50]) # print first 50 values
```

```
##  [1] 18.545030 15.892174 23.079227 17.136299 15.440534 30.317707 21.608762
##  [8]  9.803627 13.276052 14.021379 26.542450 20.344744 19.863935 18.569126
## [15] 13.485321 31.779667 23.854896  6.864510 24.735490 14.743613 15.849884
## [22] 18.178608  9.496106 13.735244 13.107422  8.710899 28.421429 17.095591
## [29] 13.549252 24.156315 24.313864 16.337555 22.293684 24.659496 23.695654
## [36] 20.873978 19.703089 14.818788 11.980750 16.015245 12.067635 15.437496
## [43] 11.111092 33.587759 29.674003 10.581239 13.866352 16.333803 27.044933
## [50] 17.437243
```

Task B) Scatter plot of Power(W) and Voltage(V)

```
# Create scatter plot
plot(voltage, power, pch = 20, col = "blue",
     xlab = "Voltage (V)", ylab = "Power (W)",
     main = "Scatter Plot - Power vs Voltage")
```



As the Voltage increases the the Power increases proportionally. As the Resistance remains constant power dissipation is high in higher voltages.

Task C) Mean and Variance of power(W)

```
# Calculate mean and variance of power
mean_power <- mean(power)
variance_power <- var(power)

# Display mean and variance of power
cat("Approximate Mean Power:", mean_power, "W\n")

## Approximate Mean Power: 18.79499 W

cat("Approximate Variance of Power:", variance_power, "W^2")

## Approximate Variance of Power: 44.06325 W^2
```

Task D) Probability of Power(W) > 20W

```
# Estimate probability that power is greater than 20 W
prob_greater_than_20 <- mean(power > 20)

# Display Estimation
cat("Probability of Power being Greater than 20 W:", prob_greater_than_20)

## Probability of Power being Greater than 20 W: 0.3835714
```

Question 3

Task A) Generate N = 500 samples (Size = 50, Uniform [-5, 5]) distribution

```
# Set seed
set.seed(123)

# Parameters
num_samples <- 500
sample_size <- 50

# Generate 500 samples of size 50 from Uniform [-5, 5] distribution
samples <- replicate(num_samples, runif(sample_size, min = -5, max = 5))

first_column <- samples[, 1]

# Check the dimensions of 'samples'
num_samples_generated <- ncol(samples)
sample_observation_count <- nrow(samples)

# Display the number of samples and the size of each sample
cat("Number of Samples Generated:", num_samples_generated, "\n")

## Number of Samples Generated: 500

cat("Size of Each Sample :", sample_observation_count,"\n")

## Size of Each Sample : 50

# Display first column of samples
cat("Fisrt column of samples:", "\n")

## Fisrt column of samples:

print(first_column) # print first 50 values

##  [1] -2.1242248  2.8830514 -0.9102308  3.8301740  4.4046728 -4.5444350
##  [7]  0.2810549  3.9241904  0.5143501 -0.4338526  4.5683335 -0.4666584
## [13]  1.7757064  0.7263340 -3.9707532  3.9982497 -2.5391227 -4.5794047
## [19] -1.7207928  4.5450365  3.8953932  1.9280341  1.4050681  4.9426978
## [25]  1.5570580  2.0853047  0.4406602  0.9414202 -2.1084026 -3.5288635
## [31]  4.6302423  4.0229905  1.9070528  2.9546742 -4.7538632 -0.2220403
## [37]  2.5845954 -2.8359206 -1.8181899 -2.6837421 -3.5719998 -0.8545366
## [43] -0.8627567 -1.3115455 -3.4755525 -3.6119394 -2.6696590 -0.3403755
## [49] -2.3402736  3.5782772
```

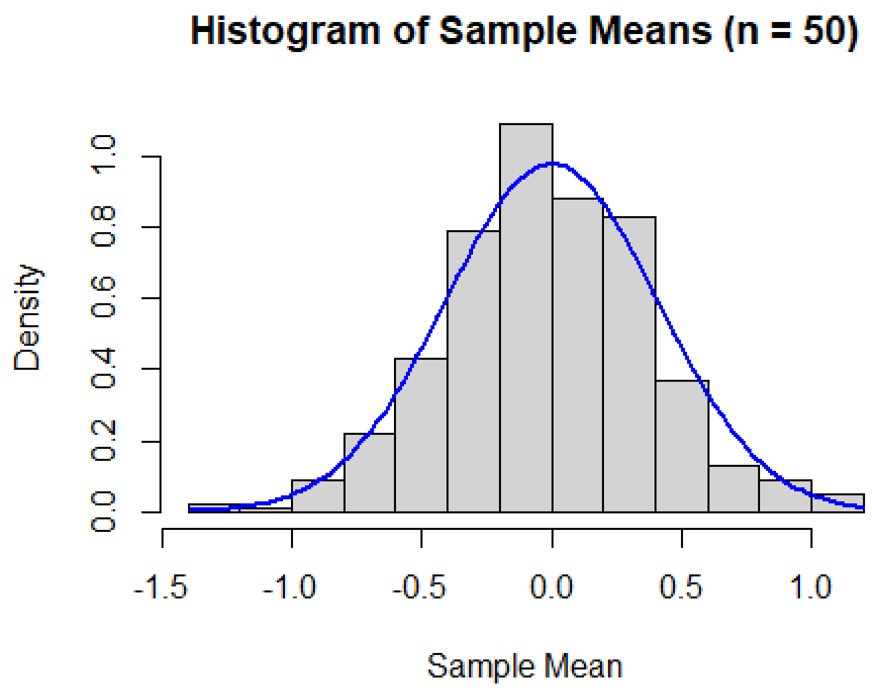
Task B) Sample mean for each sample (500 sample -> 500 sample mean)

```
# Calculate sample means for each sample
sample_means <- apply(samples, 2, mean)
print(sample_means[1:50]) # print first 50 values
```

```
##  [1]  0.2009097266 -0.2297298418  0.1538986032  0.1305989880  0.1265586016
##  [6] -0.4055247983 -0.0754521756 -0.0806986059 -0.0933522033 -0.1988394551
## [11] -0.0593896048  0.4404446346  0.2400753802 -0.1942921967  0.2048231048
## [16] -0.5081646340 -0.2887527364 -0.0007833379  0.2754145291 -0.1821835761
## [21] -0.3073738984 -0.1100968504  0.1034621153 -0.2193502777 -0.2043086532
## [26]  0.1752401245 -0.1559285703 -0.8823047719  0.2687293675  0.7212182289
## [31] -0.1134652350 -0.4736892554  0.3352868117  0.0011963015  0.3408079662
## [36]  0.4632075265  0.3259505450  0.2739682526 -0.3312901351 -0.5599604012
## [41] -0.1423421231 -0.0263406796 -0.0517085272  0.2771094282 -0.2095566820
## [46] -0.0383436508 -0.0941767330  0.0231724068 -0.4280243394 -0.2256124971
```

Task C) Plot of 500 sample mean

```
# Variance = (b - a)^2 / 12
variance_uniform <- ((5 - (-5)) ^ 2) / 12

# Plot histogram of sample means
hist(sample_means, freq = FALSE,
     main = "Histogram of Sample Means (n = 50)",
     xlab = "Sample Mean", ylab = "Density")

# Display normal distribution curve
curve(dnorm(x, mean = 0, sd = sqrt(variance_uniform/sample_size)),
      add = TRUE, col = "blue", lwd = 2)
```

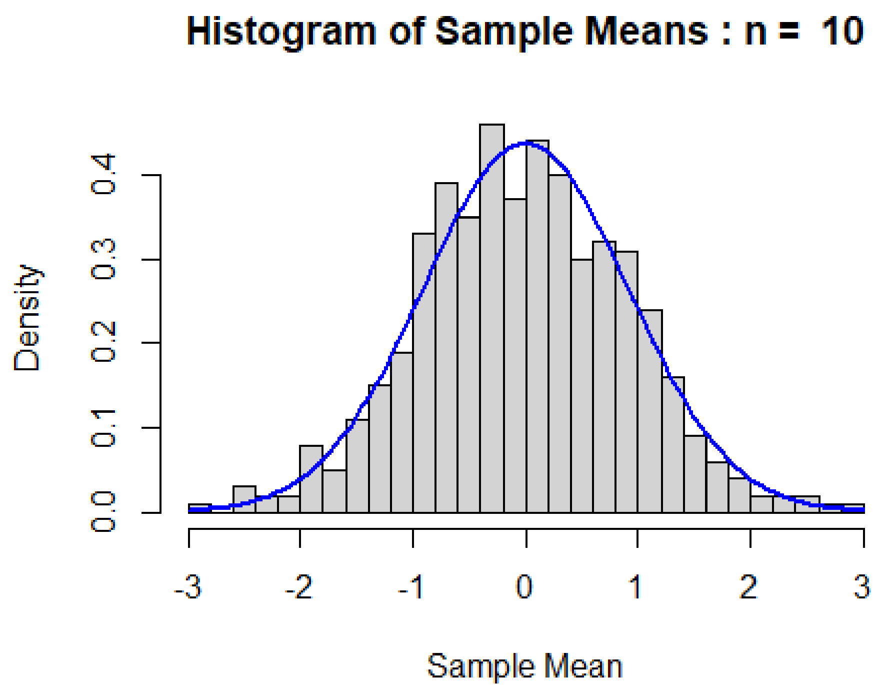


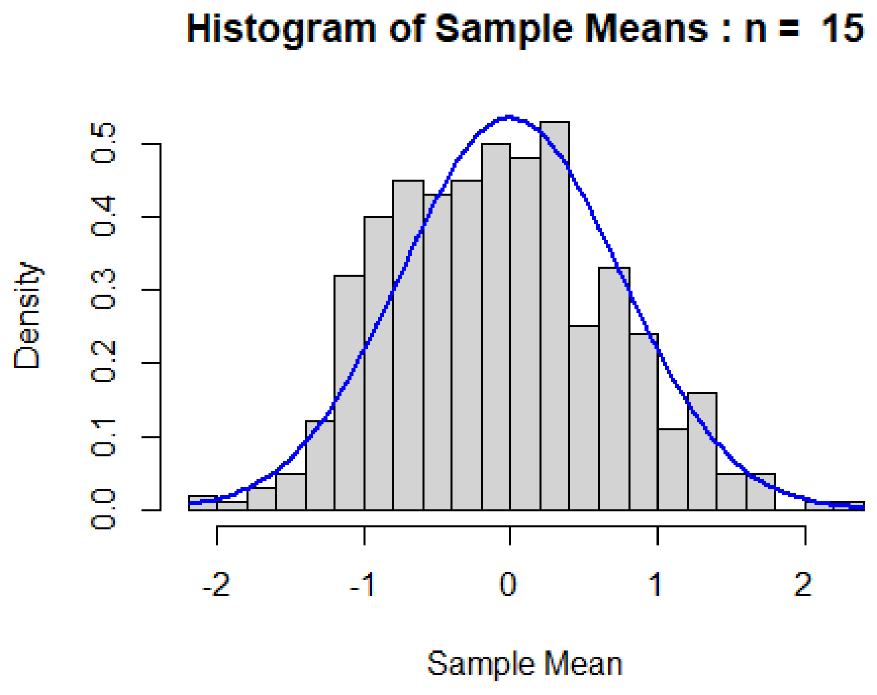
Task D) Turn simulation into function function(num_simulations, sample_size)

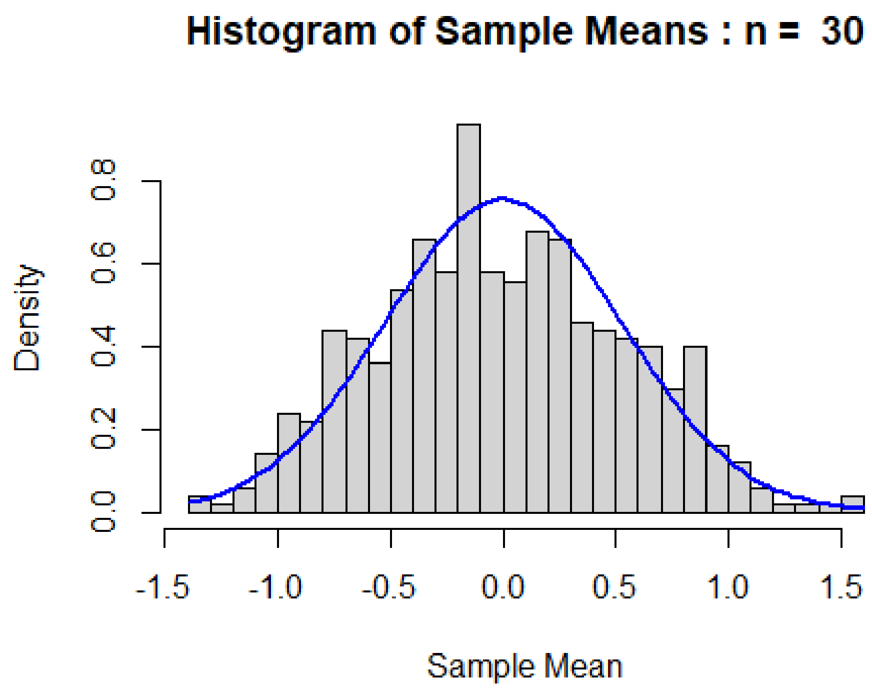
```
# Function to simulate sample means from a uniform distribution
simulate_sample_means <- function(num_simulations, sample_size) {

  # Generate samples
  samples <- replicate(num_simulations, runif(sample_size, min = -5, max =
5))

  # Calculate sample means
  sample_means <- apply(samples, 2, mean)

  return(sample_means)
}
```

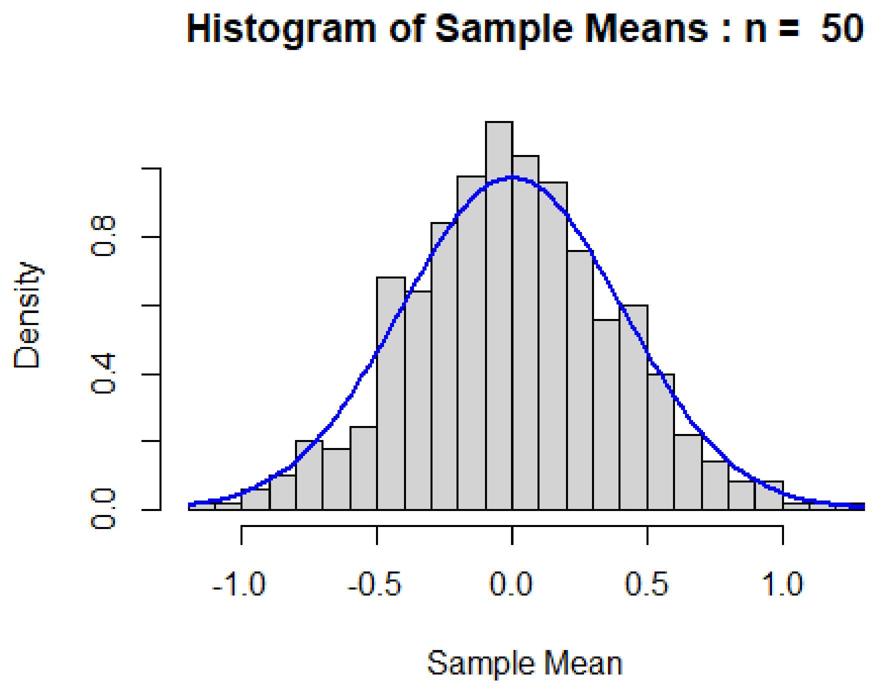
Task E) Run this function for n = 10,15,30,50.

```
# Define sample sizes
sample_sizes <- c(10, 15, 30, 50)

#par(mfrow = c(2, 2))

# Run simulation for each sample size
for (size in sample_sizes) {
  # Simulate sample means
  sample_means <- simulate_sample_means(num_simulations = 500, sample_size =
size)

  # Plot histogram of sample means
  hist(sample_means, breaks = 30, freq = FALSE,
       main = paste("Histogram of Sample Means : n = ", size),
       xlab = "Sample Mean", ylab = "Density")

  # Overlay a standard normal curve for comparison
  curve(dnorm(x, mean = 0, sd = sqrt(variance_uniform/size)),
        add = TRUE, col = "blue", lwd = 2)
}
```


Histogram of Sample Means : n = 10
Density
0.0
0.1
0.2
0.3
0.4
-3
-2
-1
0
1
2
3
Sample Mean


Histogram of Sample Means : n = 15
Density
0.0
0.1
0.2
0.3
0.4
0.5
-2
-1
0
1
2
Sample Mean



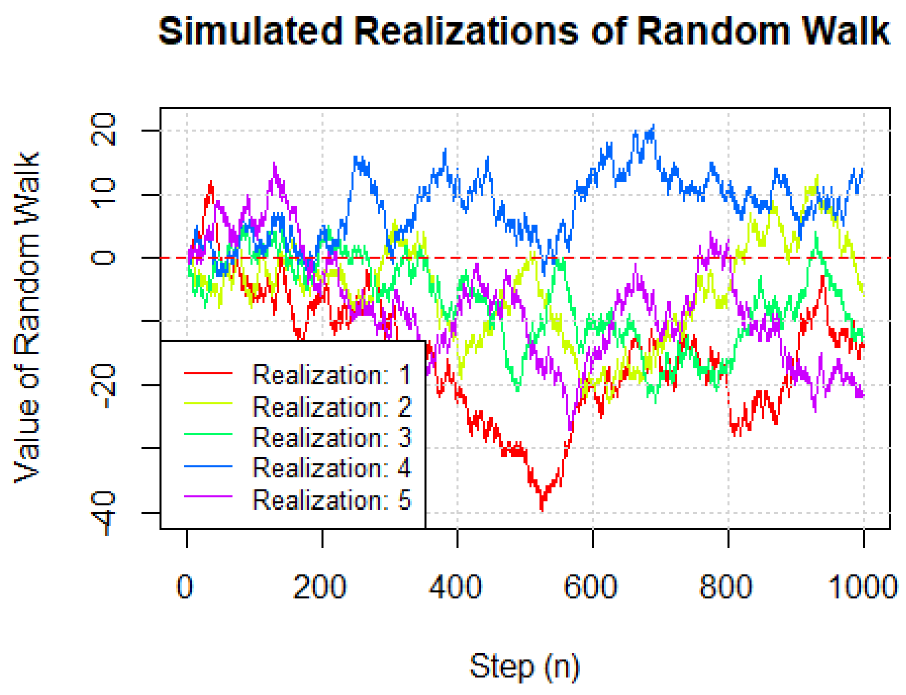


for smaller sample sizes the histogram sample means show high variability and can be different from normal distribution. as the sample size increases the sample means are becoming more symmetric and are centered around the true mean (0). in summary as the sample size increases it shows the charecteristics of central limit theorem. and in this simulation sample size 30 seems large enough.

Question 4

Task A) Simulation of 5 realization of random walk

```r
# Set seed for reproducibility
set.seed(123)

# Number of steps in the random walk
n_steps <- 1000

# Simulate five realizations of the random walk
n_realizations <- 5
random_walks <- matrix(0, nrow = n_steps, ncol = n_realizations)

for (i in 1:n_realizations) {
  # Simulate iid random variables
  X <- sample(c(1, -1), n_steps, replace = TRUE, prob = c(0.5, 0.5))

  # Calculate the random walk process
  random_walk <- cumsum(X)

  # Store the random walk in the matrix
  random_walks[, i] <- random_walk
}

# Plot the random walk realizations
matplot(1:n_steps, random_walks, type = "l", lty = 1,
        main = "Simulated Realizations of Random Walk",
        xlab = "Step (n)", ylab = "Value of Random Walk",
        col = rainbow(n_realizations))

# Add grid lines and abline at 0 to the plot
grid()
abline(h = 0, col = "red", lty = 2)

legend("bottomleft", legend =  paste("Realization:", 1:n_realizations), col =
rainbow(n_realizations), lty = 1, cex = 0.8)
```



```
# Calculate statistics for each realization
walk_statistics <- apply(random_walks, 2, function(walk) {
  list(
    mean = mean(walk),
    max = max(walk),
    min = min(walk),
    variance = var(walk)
  )
})

# Display statistics for each realization
for (i in 1:n_realizations) {
  cat("Realization:", i, "\n")
  cat("Mean:", walk_statistics[[i]]$mean, "\n")
  cat("Maximum:", walk_statistics[[i]]$max, "\n")
  cat("Minimum:", walk_statistics[[i]]$min, "\n")
  cat("Variance:", walk_statistics[[i]]$variance, "\n")
  cat("\n")
}
```

```
## Realization: 1
## Mean: -15.56
## Maximum: 12
## Minimum: -40
## Variance: 98.33273
##
## Realization: 2
## Mean: -5.054
## Maximum: 13
## Minimum: -23
## Variance: 57.76285
##
## Realization: 3
## Mean: -6.954
## Maximum: 5
## Minimum: -23
## Variance: 44.59448
##
## Realization: 4
## Mean: 7.956
## Maximum: 21
## Minimum: -3
## Variance: 25.936
##
## Realization: 5
## Mean: -7.696
## Maximum: 15
## Minimum: -27
## Variance: 73.89748
```

```
# Calculate the mean and variance of the final values of each realization
final_values <- random_walks[n_steps, ]
mean_final <- mean(final_values)
variance_final <- var(final_values)

cat("Mean of final values :", mean_final, "\n")

## Mean of final values : -8.4

cat("Variance of final values :", variance_final, "\n")

## Variance of final values : 162.8
```

Task B) Discussion of Sn as n tends to infinity

- **Mean:** The average value of the random walk over many steps tends to stay around zero.

- **Variance:** The amount of variation or spread of the random walk increases as we take more steps. With more steps, the random walk can cover a wider range of values.

- **Behavior:** As we increase the number of steps in the random walk, we see more variations or “ups and downs”. This is due to the cumulative effect of several random steps taken over time.According to a fundamental principle in probability (the Central Limit Theorem), as we take a large number of steps, the distribution of the random walk begins to resemble a bell-shaped curve, similar to a normal distribution.

- **Limiting Behavior:** As the number of steps increases, the random walk may move far from its starting location in either direction (positive or negative). The number of alternative possibilities grows without end.This behavior displays the randomness and unpredictability of the random walk process over a lengthy time.

In conclusion, when we add more steps to the random walk, it gets more unpredictable and tends to explore a wider range of values. The average location remains around zero, but the potential range of possibilities expands, illustrating the stochastic character of the random walk as we increase the number of steps.