Student ID: 112077423

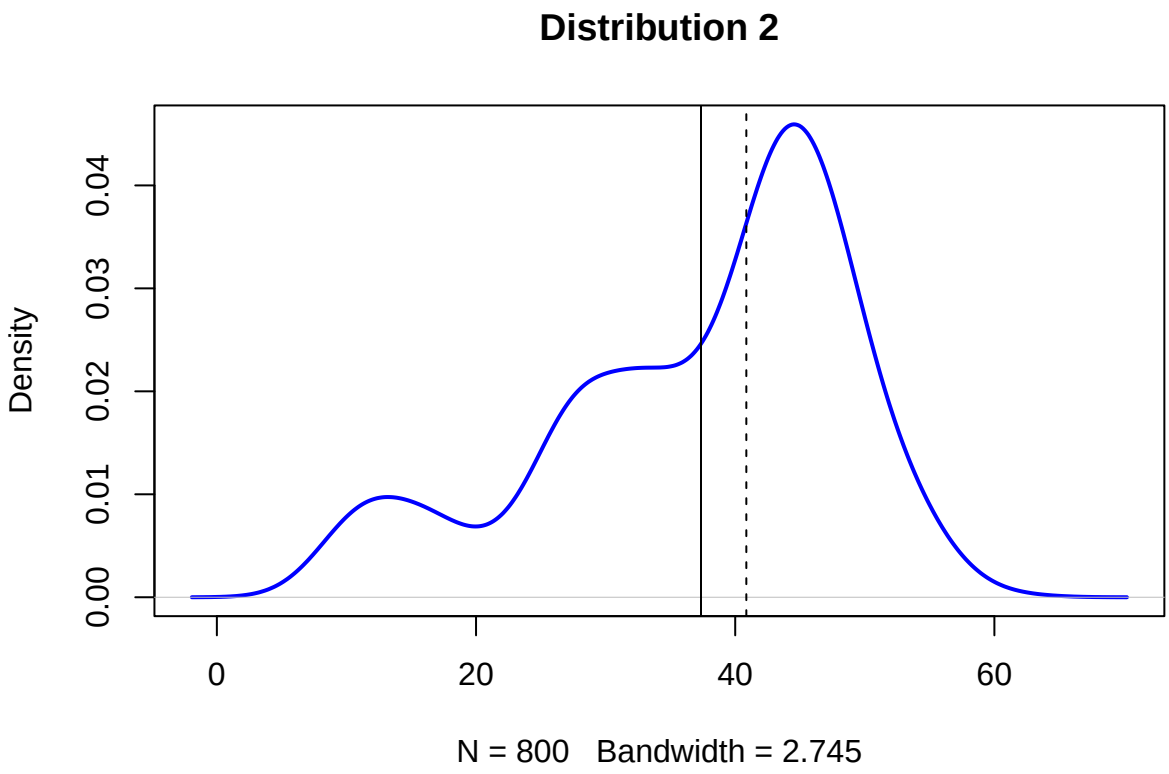
```
library(ggplot2)
library(dplyr)
```

Question 1(a)

```
# 3 normally distributed data sets
d1 <- rnorm(n=500, mean=45, sd=5)
d2 <- rnorm(n=200, mean=30, sd=4)
d3 <- rnorm(n=100, mean=15, sd=4)

d123 <- c(d1, d2, d3)

plot(density(d123), col="blue", lwd=2, main = "Distribution 2")

mean_ <- mean(d123)
median_ <- median(d123)
print(paste('mean:', mean_, '\n', 'median:', median_))
```

```
## [1] "mean: 37.3603417344809 \n median: 40.8588878888635"
```

```
# add vertical lines showing mean and median
abline(v=mean_)
abline(v=median_, lty="dashed")
```

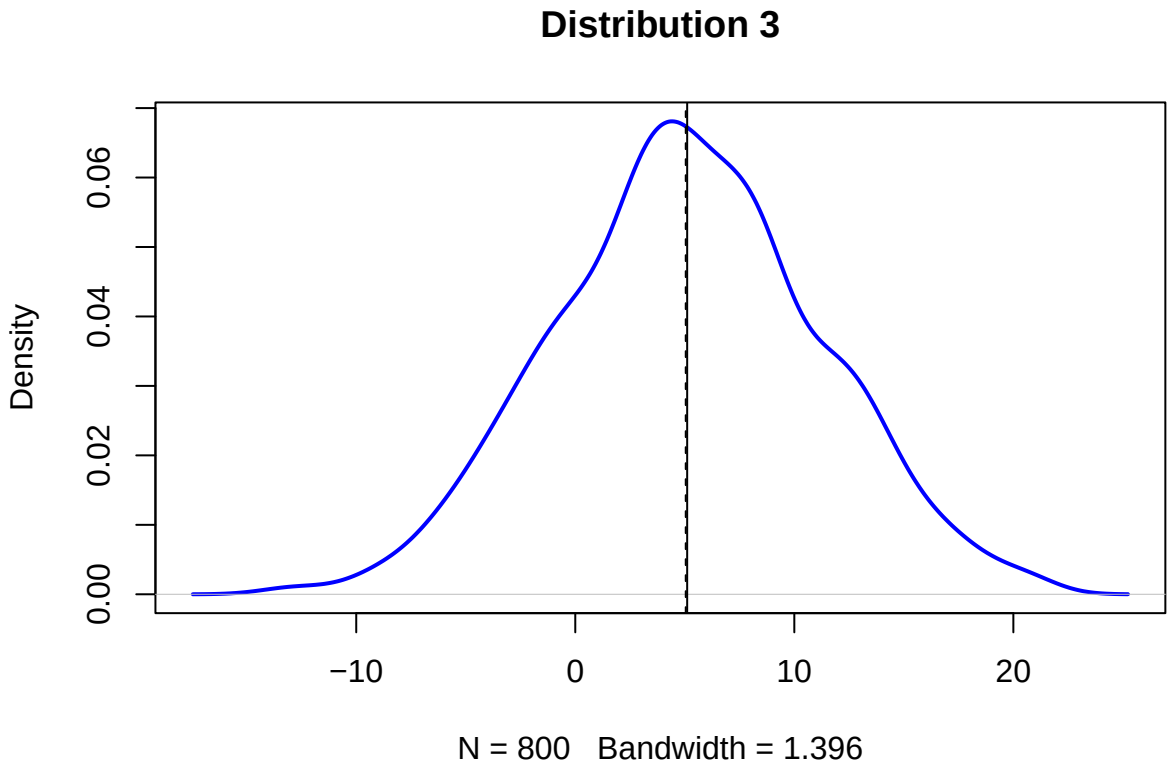


Question 1(b)

```
norm_distribution <- rnorm(n=800, mean=5, sd=6)

plot(density(norm_distribution), col='blue', lwd=2, main='Distribution 3')

mean_norm <- mean(norm_distribution)
median_norm <- median(norm_distribution)
print(paste('mean:', mean_norm, '\n', 'median:', median_norm))
```

```
## [1] "mean: 5.10892784386358 \n median: 5.0443864034989"
```

```
abline(v=mean_norm)
abline(v=median_norm, lty='dashed')
```



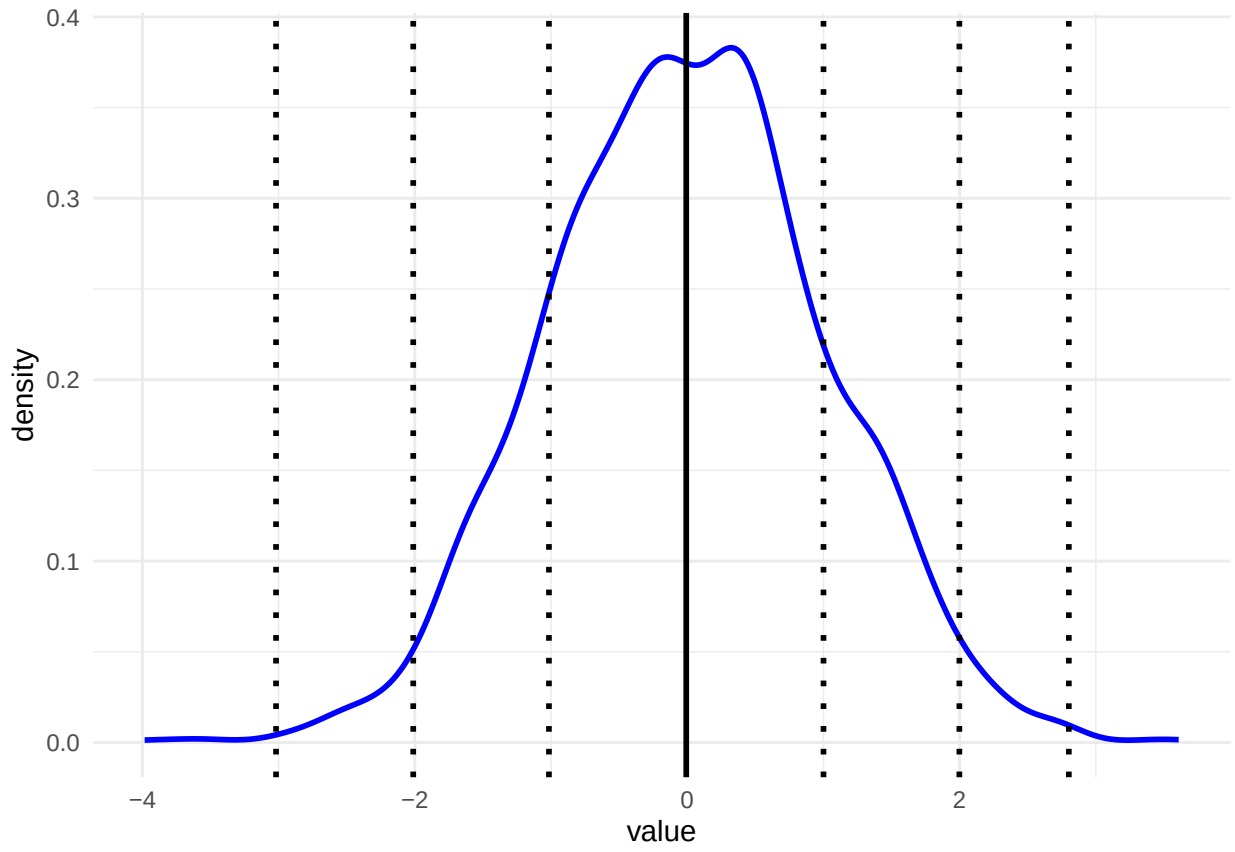
Question 1(c)

The mean is more sensitive to outliers. If large (small) numbers are added to the dataset, the mean shifts significantly, leaving us with inadequate results. On the other hand, the median is not influenced by extremely big (small) numbers, since the center of the dataset shifts slightly.

Question 2(a)

```
to_plot <- function(p, x_) {
  p <- p +
    geom_vline(xintercept = min(x_), linetype="dotted", linewidth=1) +
    geom_vline(xintercept = max(x_), linetype="dotted", linewidth=1)

  return(p)
  }

rdata <- rnorm(n=2000, mean=0, sd=1)
mean_ <- mean(rdata)
sd_ <- sd(rdata)

to_calculate_range <- function(n=1) {
  up <- mean_ + n * sd_
  low <- mean_ - n * sd_
```

```
  return(rdata[rdata > low & rdata < up])
}

p <- ggplot(rdata %>% as_tibble(), aes(value)) +
  geom_density(color='blue', lwd=1) +
  geom_vline(xintercept = mean_, linewidth = 1) +
  theme_minimal()

p <- to_plot(p, to_calculate_range())
p <- to_plot(p, to_calculate_range(2))
p <- to_plot(p, to_calculate_range(3))

p
```



Question 2(b)

```
quartiles <- quantile(rdata, c(0.25,0.5,0.75))

print(paste('The 1st quartile:', quartiles[1]))
```

```
## [1] "The 1st quartile: -0.715735752438367"
```

```r
print(paste('The 2nd quartile:', quartiles[2]))
```

```
## [1] "The 2nd quartile: -0.0115693192908644"
```

```r
print(paste('The 3rd quartile:', quartiles[3]))
```

```
## [1] "The 3rd quartile: 0.651219881618995"
```

```r
by_ <- unname((quartiles - mean_) / sd_)

print(paste(by_[1], 'standard deviations away from the mean corresponding to the 1st quartile'))
```

```
## [1] "-0.702677405259948 standard deviations away from the mean corresponding to the 1st quartile"
```

```r
print(paste(by_[2], 'standard deviations away from the mean corresponding to the 2nd quartile'))
```

```
## [1] "-0.00449062596068218 standard deviations away from the mean corresponding to the 2nd quartile"
```

```r
print(paste(by_[3], 'standard deviations away from the mean corresponding to the 3rd quartile'))
```

```
## [1] "0.652670289065007 standard deviations away from the mean corresponding to the 3rd quartile"
```

Question 2(c)

```r
new_data <- rnorm(n=2000, mean=35, sd=3.5)

quartiles <- quantile(new_data, c(0.25,0.5,0.75))
by_ <- unname((quartiles - mean(new_data)) / sd(new_data))

print(paste(by_[1], 'standard deviations away from the mean corresponding to the 1st quartile'))
```

```
## [1] "-0.696442411066815 standard deviations away from the mean corresponding to the 1st quartile"
```

```r
print(paste(by_[3], 'standard deviations away from the mean corresponding to the 3rd quartile'))
```

```
## [1] "0.680651672854841 standard deviations away from the mean corresponding to the 3rd quartile"
```

The results above are relatively close to those in (b).

Question 2(d)

```r
quartiles <- quantile(d123, c(0.25,0.5,0.75))
by_ <- unname((quartiles - mean(d123)) / sd(d123))

print(paste(by_[1], 'standard deviations away from the mean corresponding to the 1st quartile'))
```

```
## [1] "-0.64691267919408 standard deviations away from the mean corresponding to the 1st quartile"
```

```
print(paste(by_[3], 'standard deviations away from the mean corresponding to the 3rd quartile'))
```

```
## [1] "0.729678222659802 standard deviations away from the mean corresponding to the 3rd quartile"
```

The results above are relatively close to those in (b).

Question 3(a)

Rob Hyndman suggests using the Freedman–Diaconis rule (the bin-width is set to h = 2 * IQR * n^(-1/3)).
The advantage of this method is that it is less sensitive to outliers in data.

Question 3(b)

```
to_calculate_h_k <- function(dataset, n) {
  Sturges_k <- log2(n) + 1
  Scotts_h <- 3.49 * sd(dataset) / n^(1/3)
  Freedman.Diaconis_h <- 2 * IQR(dataset) / n^(1/3)
  Scotts_k = ceiling((max(dataset) - min(dataset))/Scotts_h)
  Freedman.Diaconis_k <- ceiling((max(dataset) - min(dataset))/Freedman.Diaconis_h)
  Sturges_h = (max(dataset) - min(dataset)) / Sturges_k

  return(c(Sturges_h, Sturges_k, Scotts_h, Scotts_k, Freedman.Diaconis_h, Freedman.Diaconis_k))
}
```

```
n <- 800
rand_data <- rnorm(n, mean=20, sd = 5)

results <- to_calculate_h_k(rand_data, n)

print(paste('Sturges' formula', 'h =', results[1], 'k =', results[2]))
```

```
## [1] "Sturges' formula h = 3.02671873933168 k = 10.6438561897747"
```

```
print(paste('Scott's normal reference rule', 'h =', results[3], 'k =', results[4]))
```

```
## [1] "Scott's normal reference rule h = 1.90061350358783 k = 17"
```

```
print(paste('Freedman-Diaconis' choice', 'h =', results[5], 'k =', results[6]))
```

```
## [1] "Freedman-Diaconis' choice h = 1.46459042379129 k = 22"
```

Question 3(c)

```
out_data <- c(rand_data, runif(10, min=40, max=60))
n <- 810

results <- to_calculate_h_k(out_data, n)

print(paste('Sturges' formula', 'h =', results[1], 'k =', results[2]))
```

```
## [1] "Sturges' formula h = 5.21627903659977 k = 10.661778097772"
```

```
print(paste('Scott's normal reference rule', 'h =', results[3], 'k =', results[4]))
```

```
## [1] "Scott's normal reference rule h = 2.27869911722525 k = 25"
```

```
print(paste('Freedman-Diaconis' choice', 'h =', results[5], 'k =', results[6]))
```

```
## [1] "Freedman-Diaconis' choice h = 1.48254297604845 k = 38"
```

By using Freedman-Diaconis' choice we get almost the same bin width as in (b). As mentioned before, it is because Freedman-Diaconis' choice replaces 3.5*sigma of Scott's rule with 2 IQR, which is less sensitive than the standard deviation to outliers in data.