bacs_hw2

110071010

2024-03-02

Before we start: I created a function called add_plus_sign_or_not(), intended for formatting numerical values. It would add a plus sign (+) in front of non-negative numbers and leave negative numbers unchanged.

```
add_plus_sign_or_not <- function(x) {
  formatted <- ifelse(x >= 0, paste("+", x, sep=""), as.character(x))
  return(formatted)
  }
```

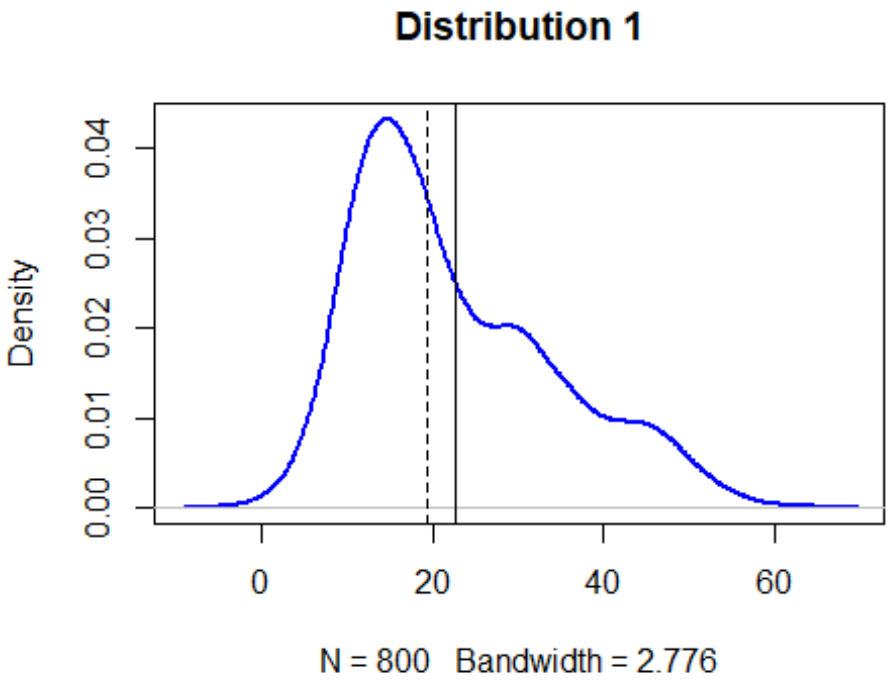
Question 1

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

# Combining them into a composite dataset
d123 <- c(d1, d2, d3)

# Plot the density function of d123
plot(density(d123), col="blue", lwd=2,
     main = "Distribution 1")

# Add vertical lines showing mean(solid) and median(dashed)
abline(v=mean(d123))
abline(v=median(d123), lty="dashed")
```



(1a)

Instructions

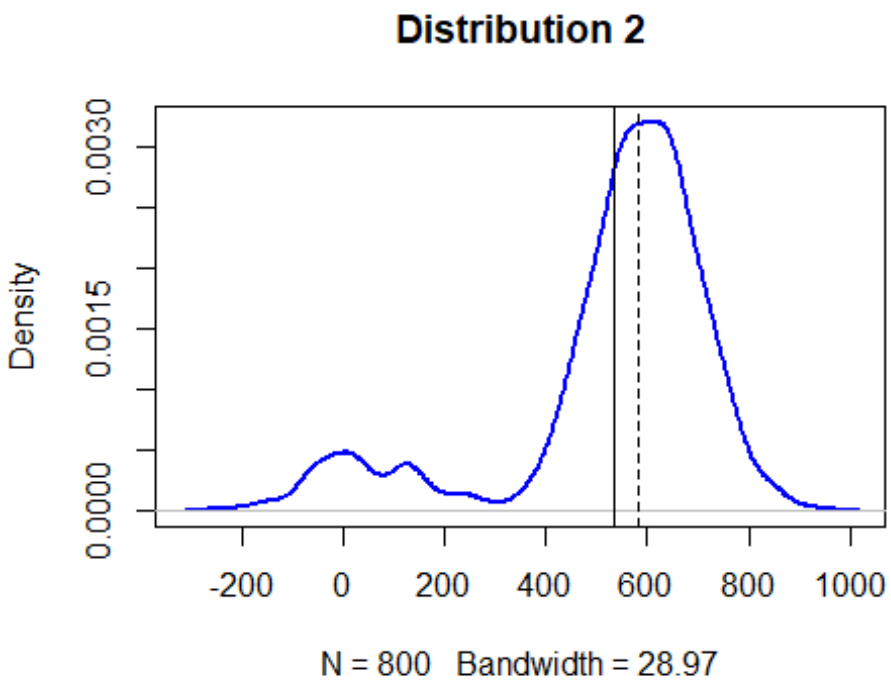
Create and visualize a new "**Distribution 2**": a combined dataset (n=800) that is negatively skewed (tail stretches to the left). Change the mean and standard deviation of d1, d2, and d3 to achieve this new distribution. Compute the mean and median, and draw lines showing the mean (solid) and median (dashed).

```
# Three normally distributed data sets, with their means and standard d
eviations being modified, though
d1 <- rnorm(n=500, mean=600, sd=100)
d2 <- rnorm(n=200, mean=600, sd=100)
d3 <- rnorm(n=100, mean=60, sd=100)

# Combining them into a composite dataset
d123 <- c(d1, d2, d3)

# Plot the density function of d123
plot(density(d123), col="blue", lwd=2,
     main = "Distribution 2")

# Add vertical lines showing mean(solid) and median(dashed)
abline(v=mean(d123))
abline(v=median(d123), lty="dashed")
```




```
# Compute and print mean and median
mean2 <- mean(d123)
median2 <- median(d123)
print("Distribution 2")

## [1] "Distribution 2"

cat("Mean:", mean2, "\n")

## Mean: 532.3595

cat("Median:", median2, "\n")

## Median: 583.4548
```

Comment

The means and standard deviations of d1, d2, and d3 have been changed to build a composite dataset. As shown in the graph, the dataset has a negatively skewed distribution; the median (582.3347) has proven to be greater than the mean (533.9848) .

(1b)

Instructions

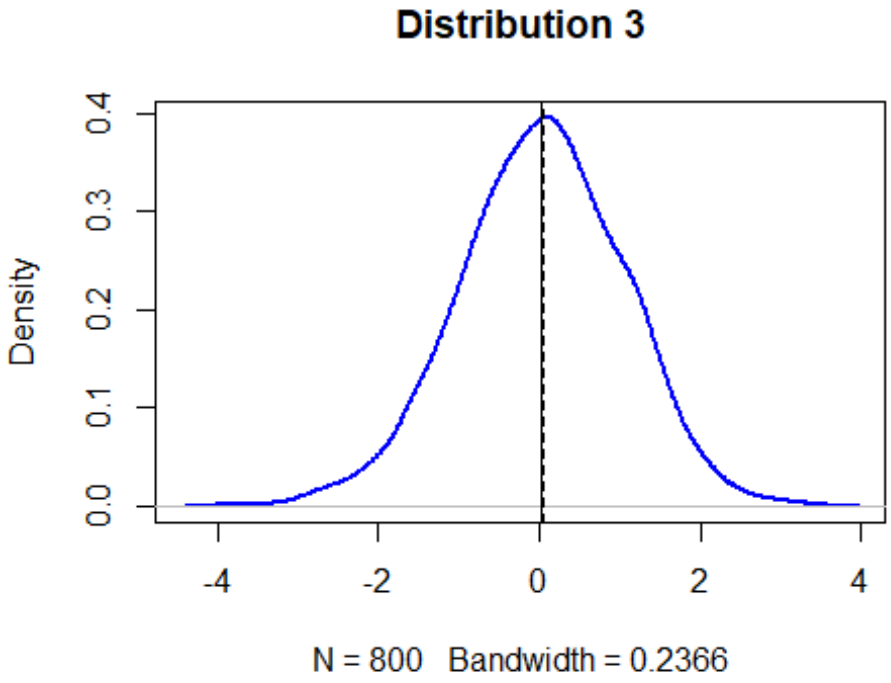
Create a "**Distribution 3**": a single dataset that is normally distributed (bell-shaped, symmetric) – you do not need to combine datasets, just use the rnorm() function to

create a single large dataset (n=800). Show your code, compute the mean and median, and draw lines showing the mean (solid) and median (dashed).

```
# Use rnorm() to simulate data with the size of 800
d4 <- rnorm(800)

# Plot the density function of d4
plot(density(d4), col="blue", lwd=2,
     main = "Distribution 3")

# Add vertical lines showing mean(solid) and median(dashed)
abline(v=mean(d4))
abline(v=median(d4), lty="dashed")
```




```
# Compute and print mean and median
mean3 <- mean(d4)
median3 <- median(d4)
print("Distribution 3")

## [1] "Distribution 3"

cat("Mean:", mean3, "\n")

## Mean: 0.02668559

cat("Median:", median3, "\n")

## Median: 0.05079791
```

Comment

According to the plot, the dataset seems to have a bell-shaped, symmetric distribution given that it is generated using rnorm() ; the median (0.002907356) has proven to be pretty close to the mean (0.01199789).

(1c)

Question

In general, which measure of central tendency (mean or median) do you think will be more sensitive (will change more) to outliers being added to your data?

My Answer

The mean is more sensitive to outliers because the process of calculating the mean involves summing all values in the dataset and then dividing by the number of observations. This process inherently gives equal weight to every data point, including outliers, which can tilt the result significantly.

On the other hand, the median, the middle value of a dataset when it is sorted from least to greatest, does not directly involve the magnitude of the values within its calculation.

Question 2

(2a)

Instructions

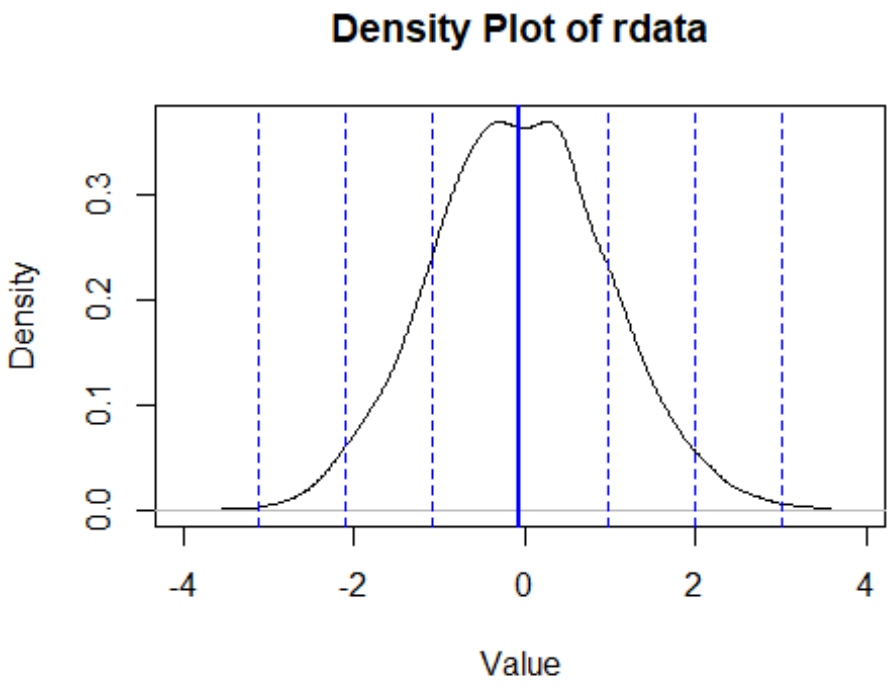
Create a random dataset (call it rdata) that is normally distributed with: n=2000, mean=0, sd=1. Draw a density plot and put a solid vertical line on the mean, and dashed vertical lines at the 1st, 2nd, and 3rd standard deviations to the left and right of the mean. You should have a total of 7 vertical lines (one solid, six dashed).

```
# Randomly generate a normally-distributed dataset (n = 2000) by rnorm
(). Note that rnorm() has the default mean value of 0 and sd of 1
rdata <-rnorm(2000)

# Compute the mean and sd of rdata
mean_rdata <- mean(rdata)
sd_rdata <- sd(rdata)

# Plot the density function of rdata
plot(density(rdata), main="Density Plot of rdata", xlab="Value", ylab="
Density")
abline(v=mean_rdata, col="blue", lwd=2)
```

```
# Place the lines of mean and 1st, 2nd, and 3rd standard deviations to
the left and right of the mean
sd_positions <- c(-3, -2, -1, 1, 2, 3) * sd_rdata + mean_rdata
abline(v=sd_positions, lty="dashed", col="blue")
```



(2b)

Instructions

Using the quantile() function, which data points correspond to the 1st, 2nd, and 3rd quartiles (i.e., 25th, 50th, 75th percentiles) of rdata? How many standard deviations away from the mean (divide by standard-deviation; keep positive or negative sign) are those points corresponding to the 1st, 2nd, and 3rd quartiles?

```
print("rdata")
```

```
## [1] "rdata"
```

```
# Compute how many std away the quartiles are
result_2b <- (quantile(rdata)-mean(rdata))/sd(rdata)
# Keep the positive or negative sign by predefined add_plus_sign_or_not
()
add_plus_sign_or_not(round(result_2b, 5))
```

```
##         0%        25%        50%        75%       100%
## "-3.28529" "-0.69008" "-0.01279" "+0.65536" "+3.29603"
```

(2c)

Instructions

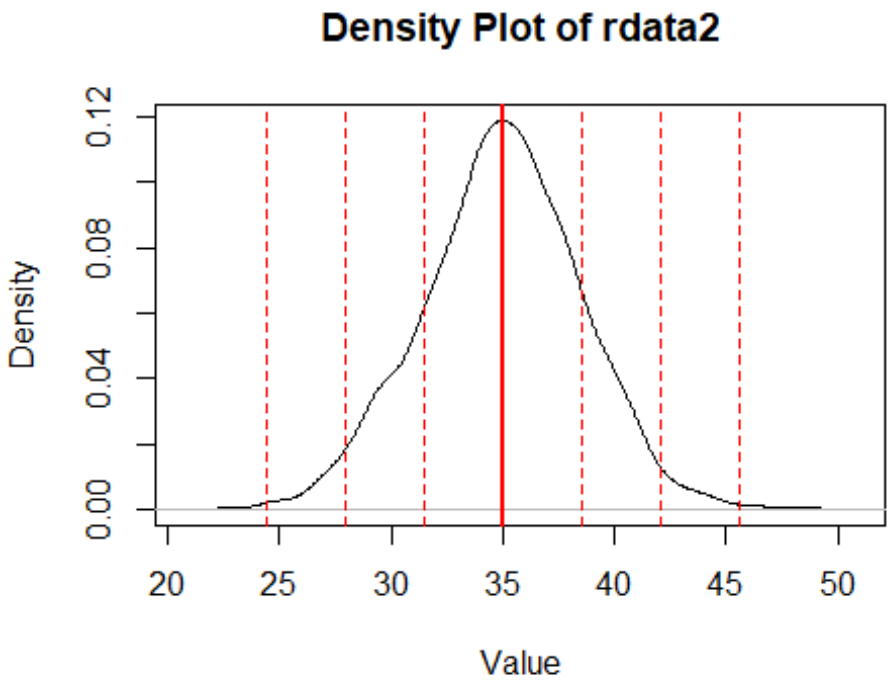
Now create a new random dataset that is normally distributed with: n=2000, mean=35, sd=3.5.
In this distribution, how many standard deviations away from the mean (use positive or negative) are those points corresponding to the 1st and 3rd quartiles? Compare your answer to (b)

```
rdata2 <-rnorm(2000, mean=35, sd= 3.5)

# Compute the mean and sd of rdata2
mean_rdata2 <- mean(rdata2)
sd_rdata2 <- sd(rdata2)

# Plot the density function of rdata2
plot(density(rdata2), main="Density Plot of rdata2", xlab="Value", ylab
="Density")
abline(v=mean_rdata2, col="red", lwd=2)

# Place the lines of mean and 1st, 2nd, and 3rd standard deviations to
the left and right of the mean
sd_positions <- c(-3, -2, -1, 1, 2, 3) * sd_rdata2 + mean_rdata2
abline(v=sd_positions, lty="dashed", col="red")
```



```
print("rdata2")
```

```
## [1] "rdata2"
```

```
# Compute how many std away the quartiles are
result_2c <- (quantile(rdata2)-mean(rdata2))/sd(rdata2)
# Keep the positive or negative sign by predefined add_plus_sign_or_not
()
add_plus_sign_or_not(round(result_2c, 5))
```

```
##         0%        25%        50%        75%       100%
## "-3.48095" "-0.62921" "+0.01621" "+0.66595" "+3.94354"
```

Comparing to (2b)

```
# Calculate quartiles for rdata
quartiles_rdata <- quantile(rdata, probs = c(0.25, 0.75))
# Standard deviations away from the mean for each quartile in rdata
sd_away_rdata <- (quartiles_rdata - mean_rdata) / sd_rdata

# Calculate quartiles for rdata2
quartiles_rdata2 <- quantile(rdata2, probs = c(0.25, 0.75))
# Standard deviations away from the mean for each quartile in rdata2
sd_away_rdata2 <- (quartiles_rdata2 - mean_rdata2) / sd_rdata2

# Printing results
sd_away_rdata <- add_plus_sign_or_not(round(sd_away_rdata, 5))
sd_away_rdata2<- add_plus_sign_or_not(round(sd_away_rdata2, 5))
cat("SDs away from mean for rdata (1st and 3rd quartiles):", sd_away_rd
ata, "\n")
```

```
## SDs away from mean for rdata (1st and 3rd quartiles): -0.69008 +0.65
536
```

```
cat("SDs away from mean for rdata2 (1st and 3rd quartiles):", sd_away_r
data2, "\n")
```

```
## SDs away from mean for rdata2 (1st and 3rd quartiles): -0.62921 +0.6
6595
```

(2d)

Instructions

Finally, recall the dataset d123 shown in the description of question 1. In that distribution, how many standard deviations away from the mean (use positive or negative) are those data points corresponding to the 1st and 3rd quartiles? Compare your answer to (b)

```
print("d123")
```

```
## [1] "d123"
```

```r
# Compute how many std away the quartiles are
result_2d <- (quantile(d123)-mean(d123))/sd(d123)
# Keep the positive or negative sign by predefined add_plus_sign_or_not
()
add_plus_sign_or_not(round(result_2d, 5))
```

```
##         0%        25%        50%        75%       100%
## "-3.65304" "-0.20312" "+0.24407" "+0.58145" "+1.90033"
```

```r
# Calculate quartiles for d123
quartiles_d123 <- quantile(d123, probs = c(0.25, 0.75))
# Standard deviations away from the mean for each quartile in rdata2
sd_away_d123 <- (quartiles_d123 - mean(d123))/ sd(d123)
sd_away_d123 <- add_plus_sign_or_not(round(sd_away_d123, 5))

# Printing results
cat("SDs away from mean for rdata (1st and 3rd quartiles):", sd_away_rd
ata, "\n")
```

```
## SDs away from mean for rdata (1st and 3rd quartiles): -0.69008 +0.65
536
```

```r
cat("SDs away from mean for d123 (1st and 3rd quartiles):", sd_away_d12
3, "\n")
```

```
## SDs away from mean for d123 (1st and 3rd quartiles): -0.20312 +0.581
45
```

Question 3

(3a)

Question

From the question on the forum, which formula does Rob Hyndman's answer (1st answer) suggest to use for bin widths/number? Also, what does the Wikipedia article say is the benefit of that formula?

My Answer

i. Bin widths (h) = 2 × IQR × n^(−1/3), and the number of bins (k) = ceiling((max−min)/h)

ii. 2 * IQR in Freedman–Diaconis' choice is less sensitive than the standard deviation is to outliers in data.

(3b)

Instructions

Given a random normal distribution:
rand_data <- rnorm(800, mean=20, sd = 5)

Compute the bin widths (h) and number of bins (k) according to each of the following formula:
i. Sturges' formula
ii. Scott's normal reference rule (uses standard deviation)
iii. Freedman-Diaconis' choice (uses IQR)

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

# Number of observations
n <- length(rand_data)

# Range of the data
data_range <- max(rand_data) - min(rand_data)

# (i) Sturges' formula
k_sturges <- ceiling(1 + log2(n))
h_sturges <- data_range / k_sturges

# (ii) Scott's normal reference rule
sd_rand_data <- sd(rand_data)
h_scott <- 3.49 * sd_rand_data / (n^(1/3))
k_scott <- data_range / h_scott

# (iii) Freedman-Diaconis' choice
iqr_rand_data <- IQR(rand_data)
h_freedman <- 2 * iqr_rand_data / (n^(1/3))
k_freedman <- data_range / h_freedman

# Print results
cat("Sturges' formula: k =", k_sturges, ", h =", h_sturges, "\n")

## Sturges' formula: k = 11 , h = 2.946988

cat("Scott's rule: k =", k_scott, ", h =", h_scott, "\n")

## Scott's rule: k = 17.46693 , h = 1.855899

cat("Freedman-Diaconis' choice: k =", k_freedman, ", h =", h_freedman,
"\n")

## Freedman-Diaconis' choice: k = 22.74771 , h = 1.425061
```

(3c)

Instructions

Repeat part (b) but let's extend rand_data dataset with some outliers (creating a new dataset out_data):
out_data <- c(rand_data, runif(10, min=40, max=60))

```
# Inserting outliers into the original dataset
out_data <- c(rand_data, runif(10, min=40, max=60))

# Number of observations
n <- length(out_data)

# Range of the data
data_range <- max(out_data) - min(out_data)

# (i) Sturges' formula
k_sturges2 <- ceiling(1 + log2(n))
h_sturges2 <- data_range / k_sturges2

# (ii) Scott's normal reference rule
sd_out_data <- sd(out_data)
h_scott2 <- 3.49 * sd_out_data/ (n^(1/3))
k_scott2 <- data_range / h_scott2

# (iii) Freedman-Diaconis' choice
iqr_out_data <- IQR(out_data)
h_freedman2 <- 2 * iqr_out_data / (n^(1/3))
k_freedman2 <- data_range / h_freedman2

# Print results
print("out_data")

## [1] "out_data"

cat("Sturges' formula: k =", k_sturges2, ", h =", h_sturges2, "\n")

## Sturges' formula: k = 11 , h = 4.824374

cat("Scott's rule: k =", k_scott2, ", h =", h_scott2, "\n")

## Scott's rule: k = 23.85497 , h = 2.224615

cat("Freedman-Diaconis' choice: k =", k_freedman2, ", h =", h_freedman2,
 "\n")

## Freedman-Diaconis' choice: k = 36.95819 , h = 1.435896
```

110071010

Question

From your answers above, in which of the three methods does the bin width (h) change the least when outliers are added (i.e., which is least sensitive to outliers), and (briefly) WHY do you think that is?

My Answer

```
# Computing the bin width differences
hdiff_sturge <- abs(h_sturges - h_sturges2)
hdiff_scott <- abs(h_scott - h_scott2)
hdiff_freedman <- abs(h_freedman - h_freedman2)

print("Bin width differences")

## [1] "Bin width differences"

cat("Sturges' formula:" , hdiff_sturge, "\n")

## Sturges' formula: 1.877386

cat("Scott's rule:" , hdiff_scott, "\n")

## Scott's rule: 0.3687156

cat("Freedman-Diaconis:" , hdiff_freedman, "\n")

## Freedman-Diaconis: 0.01083495
```

The result demonstrates that, among the three methods, Freedman-Diaconis' choice is the least affected by the outliers. The reason lies in the fact that it is the only one that adopts IQR, which measures the middle 50 % of the data. The design of IQR itself has helped prevent the result from being significantly influenced by the outliers.