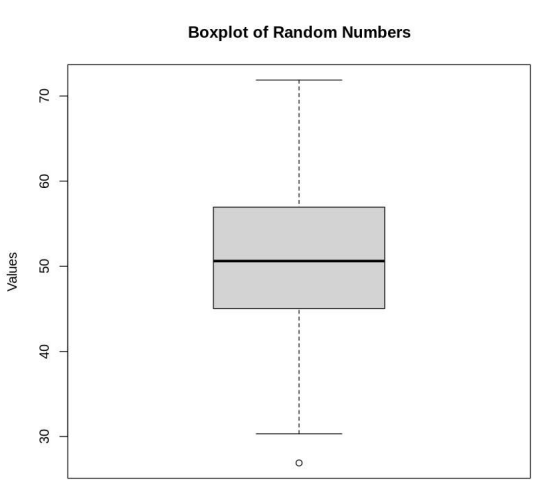
```
#Generate 100 random numbers and find out outliers among them

# Generate 100 random numbers
set.seed(123) # for reproducibility
numbers <- rnorm(100, mean = 50, sd = 10)

# Create a boxplot to visualize the data and identify outliers
boxplot(numbers, main = "Boxplot of Random Numbers", ylab = "Values")
```



This will create a boxplot of the random numbers, with a box that represents the interquartile range (IQR) of the data (the middle 50% of the data). Any data points that fall outside of the IQR can be considered outliers. These will be represented as individual points outside of the box in the boxplot.

```
#Tukey Method
outliers <- boxplot.stats(numbers)$out
print(outliers)
```

```
[1] 26.90831
```

Tukey method to find outliers, which defines outliers as any value that is more than 1.5 times the IQR below the first quartile or above the third quartile.

```
#Z-score
z_scores <- scale(numbers)
outliers <- which(abs(z_scores) > 3)
print(outliers)
```

```
integer(0)
```

The z-score method calculates the number of standard deviations a data point is from the mean. Data points with a z-score of more than 3 or less than -3 are considered outliers.

```
#Modified Z-score
median <- median(numbers)
mad <- median(abs(numbers - median))
modified_z_scores <- 0.6745 * (numbers - median) / mad
outliers <- which(abs(modified_z_scores) > 3)
print(outliers)
```

```
integer(0)
```

The modified z-score method is similar to the z-score method, but uses a more robust estimate of the scale of the data (the median absolute deviation) instead of the standard deviation.

```
#Grubb's Test
install.packages("outliers")
library(outliers)
grubbs.test(numbers)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```

        Grubbs test for one outlier

data:  numbers
G = 2.62876, U = 0.92949, p-value = 0.3792
alternative hypothesis: lowest value 26.9083112435919 is an outlier
```

Grubbs' test is a statistical test that can be used to identify a single outlier in a univariate dataset. It is based on the assumption that the data are normally distributed.

```
# Calculate mean and standard deviation of data
mean <- mean(numbers)
sd <- sd(numbers)

# Define a function to calculate Mahalanobis distance
mahalanobis <- function(x, mean, sd) {
  return(sqrt((x - mean)^2/sd^2))
}

# Calculate Mahalanobis distance for each data point
mahal_dist <- sapply(numbers, mahalanobis, mean = mean, sd = sd)

# Identify outliers using a threshold for Mahalanobis distance
threshold <- 3 # adjust this value as needed
outliers <- which(mahal_dist > threshold)

# Print the outliers
print(outliers)
```

```
integer(0)
```

Mahalanobis distance method calculates the distance of each data point from the mean of the dataset taking into account the covariance of the data.

It's worth noting that each method can have different assumptions and may give different results, so it's important to consider the underlying assumptions and characteristics of the data when choosing a method.