Berkstats Day 1: First Steps in R

Ulrich Matter

2/19/2017

The R-Console

```
print("Hello world")
```

```
## [1] "Hello world"
```

Variables and Vectors

```
a <- c(10,22,33, 22, 40)
names(a) <- c("Andy", "Betty", "Claire", "Daniel", "Eva")
a
```

```
##   Andy  Betty Claire Daniel    Eva
##     10     22     33     22     40
```

```
a[3]
```

```
## Claire
##     33
```

```
a["Claire"]
```

```
## Claire
##     33
```

Compute the Mean

```
# own implementation: use R-function for summing up the elements in a vector
# and getting the number of elements in a vector
sum(a) / length(a)
```

```
## [1] 25.4
```

```
# or use the function delivered with the R installation
mean(a)
```

```
## [1] 25.4
```

Compute the Median

```
# own implementation
# 1) sort the vector in ascending order (if not yet ordered)
sorted_a <- sort(a)
```

```r
# 2) get the index of the element in the middle (i.e., the [(N + 1)/2]th element)
middle <- (length(sorted_a)+1)/2
# 3) check whether the index of the element we get is a fraction
is_fraction <- (middle %% 1) != 0
# 4) if so, take the mean of the element above and below as median
#    else, take that middle element as median
if (is_fraction) {
     (sorted_a[floor(middle)] + sorted_a[ceiling(middle)]) / 2
} else {
     sorted_a[middle]
}
```

```
## Daniel
##     22
```

```r
# function delivered with R-installation
median(a)
```

```
## [1] 22
```

Compute the Mode

```r
# count occurrences
counts <- table(a)
# which value occurs most often
which.max(counts)
```

```
## 22
##  2
```

```r
# write your own mode-function:
mymode <- function(x) {
     counts <- table(x)
     x_mode <- as.numeric(names(which.max(counts)))

     return(x_mode)
}

# apply your own mode-function:
mymode(a)
```

```
## [1] 22
```

Measures of Variability

```r
range(a)
```

```
## [1] 10 40
```

```r
var(a)
```

```
## [1] 132.8
```

```
sd(a)
```

```
## [1] 11.52389
```

Compute the Standard Deviation

```
# own implementaion
sqrt(sum((a-mean(a))^2) / (length(a) - 1))
```

```
## [1] 11.52389
```

```
# function delivered with R-installation
sd(a)
```

```
## [1] 11.52389
```

Random Draws and Distributions

```
normal_distr <- rnorm(1000)
hist(normal_distr)
```

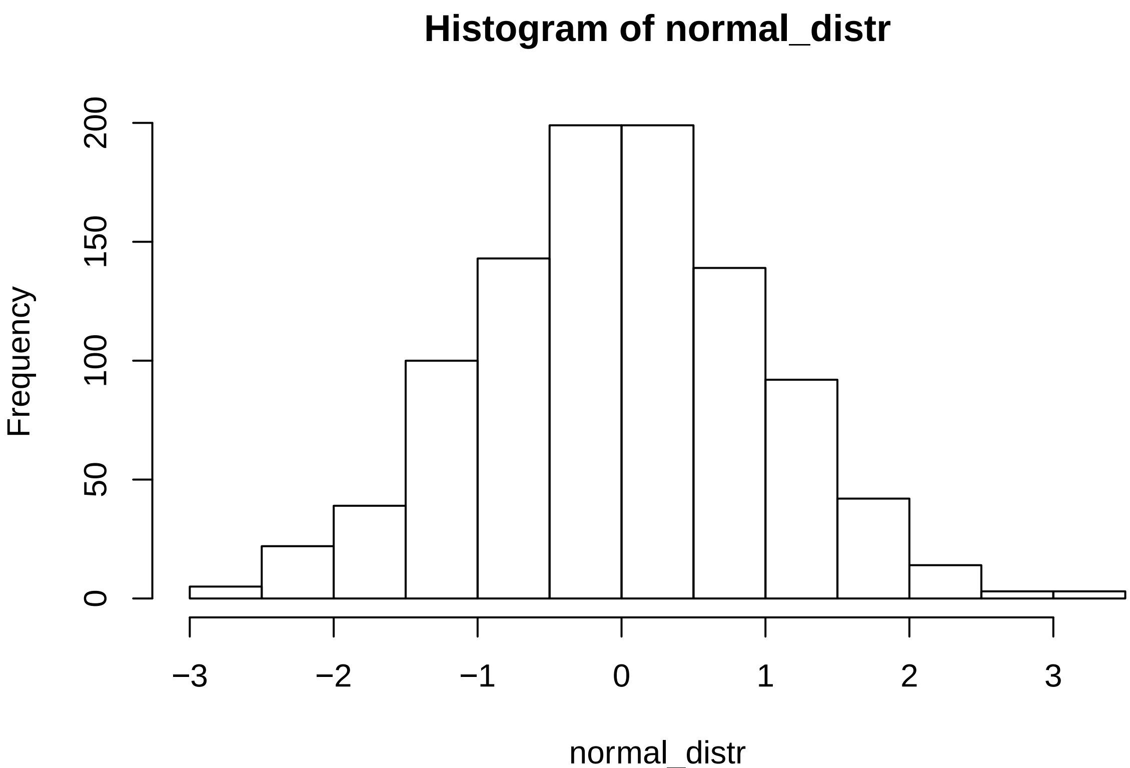
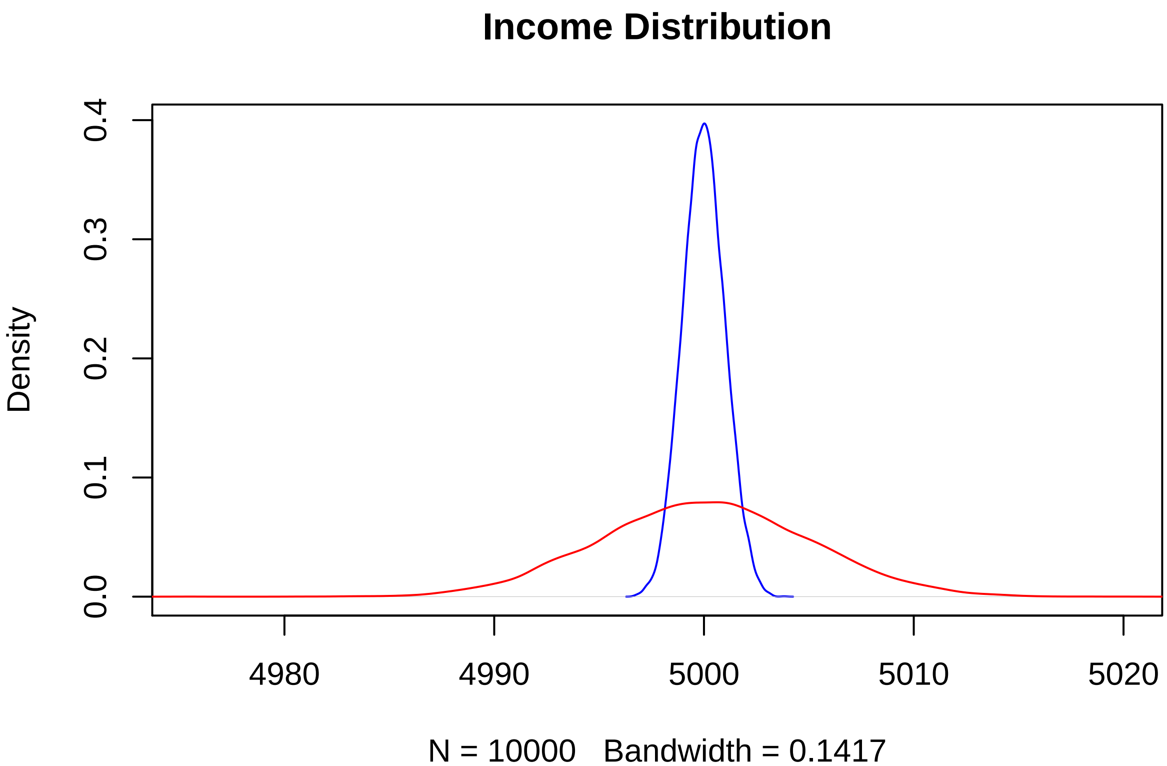


Illustration of Variability

```r
# draw a random sample from a normal distribution with a large standard deviation
largevar <- rnorm(10000, mean = 5000, sd = 5)
# draw a random sample from a normal distribution with a small standard deviation
littlevar <- rnorm(10000, mean = 5000, sd = 1)

# visualize the distributions of both samples with a density plot
plot(density(littlevar), col = "blue",
     xlim=c(min(largevar), max(largevar)), main="Income Distribution")
lines(density(largevar), col = "red")
```



Note: the red curve illustrates the distribution of the sample with a large standard deviation (a lot of variability) whereas the blue curve illustrates the one with a rather small standard deviation.

Skewness and Kurtosis

```r
# Install the R-package called "moments" with the following command (if not installed yet):
# install.packages("moments")

# load the package
library(moments)
```

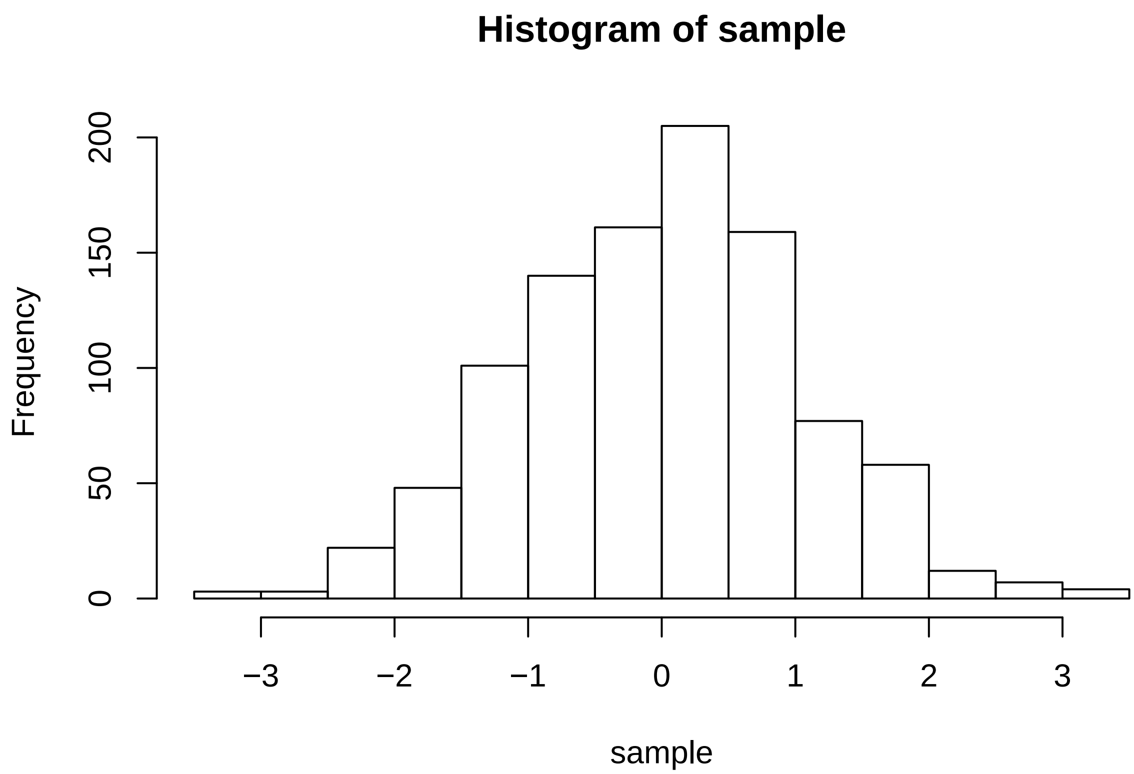
Recall Day 1's slides on Skewness and Kurtosis. A helpful way to memorize what a certain value of either of these two statistics means is to visualize the respective distribution (as shown in the slides).

Skewness

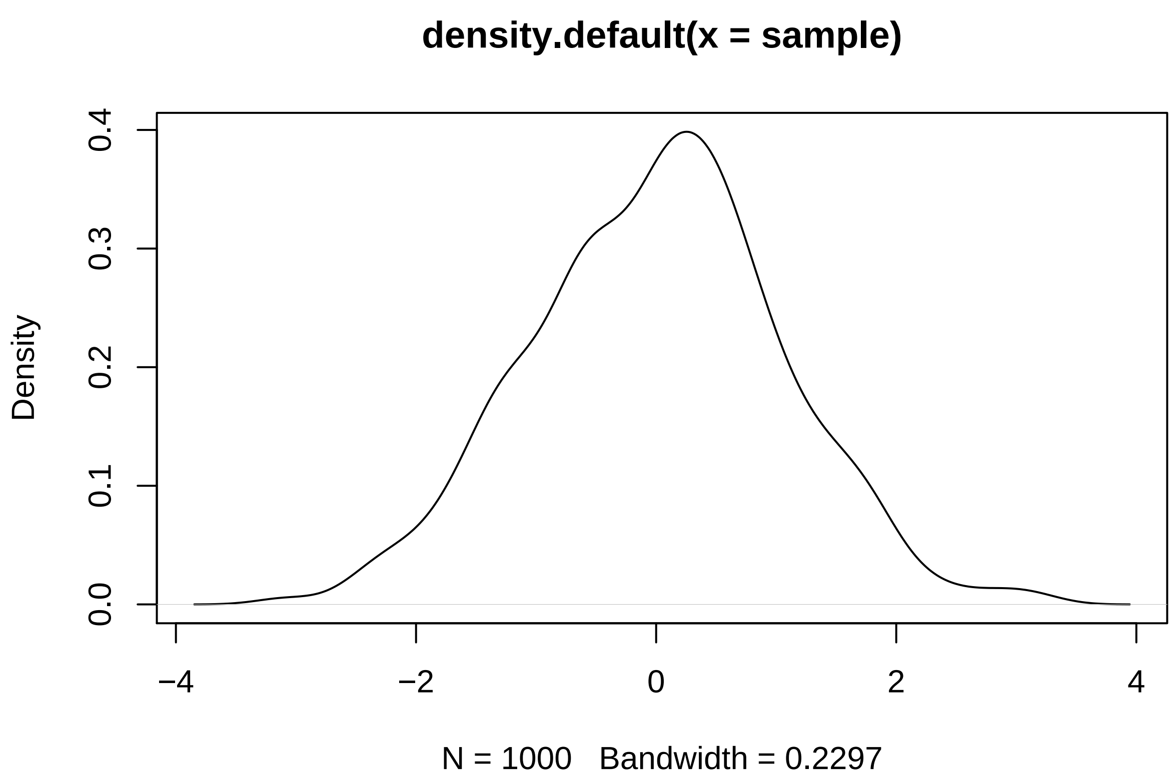
Skewness refers to how symetric the frequency distribution of a variable is. For example, a distribution can be 'positively skewed' meaning it has a long tail on the right and a lot of 'mass' (observations) on the left. We can see that when visualizing the distribution in a histogram or a density plot. Lets have a look at this in R (note the comments in the code explaing what each line does):

```r
# draw a random sample of simulated data from a normal distribution
# the sample is of size 1000 (hence, n = 1000)
sample <- rnorm(n = 1000)

# plot a histogram and a density plot of that sample
# note that the distribution is neither strongly positively nor negatively skewed
# (this is to be expected, as we have drawn a sample from a normal distribution!)
hist(sample)
```




```r
plot(density(sample))
```

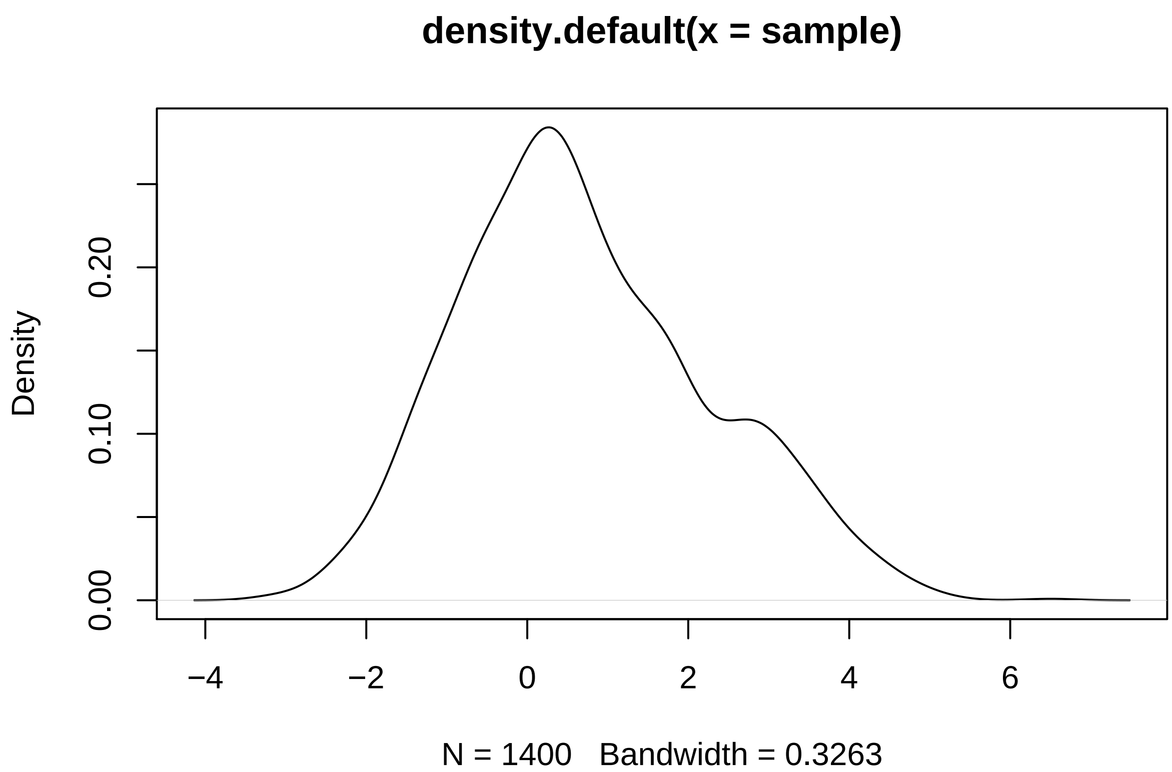



```r
# now compute the skewness
skewness(sample)
```

```
## [1] 0.005174825
```

```r
# Now we intentionally change our sample to be strongly positively skewed
# We do that by adding some outliers (observations with very high values) to the sample
sample <- c(sample, (rnorm(200) + 2), (rnorm(200) + 3))

# Have a look at the distribution and re-calculate the skewness
plot(density(sample))
```




```r
skewness(sample)
```

```
## [1] 0.4038113
```

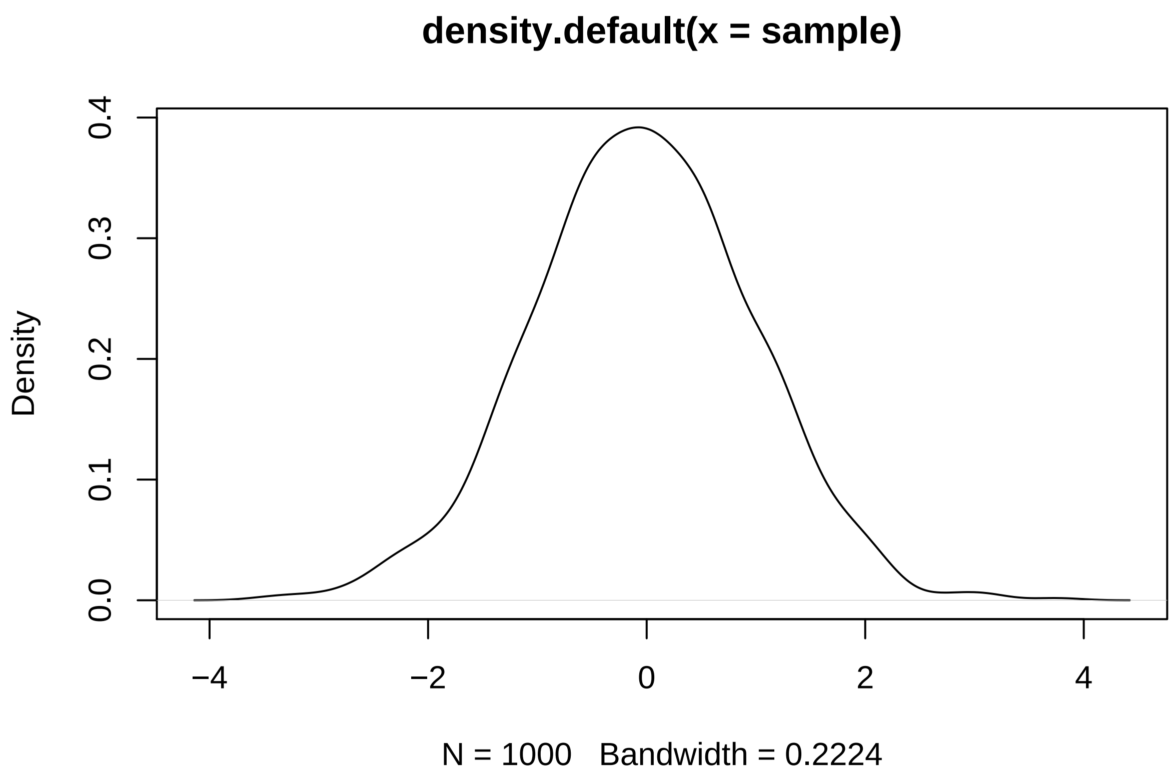
```r
#
```

Kurtosis

Kurtosis refers to how much 'mass' a distribution has in its 'tails'. It thus tells us something about whether a distribution tends to have a lot of outliers. Again, plotting the data can help us understand this concept of kurtosis. Lets have a look at this in R (note the comments in the code explaing what each line does):

```r
# draw a random sample of simulated data from a normal distribution
# the sample is of size 1000 (hence, n = 1000)
sample <- rnorm(n = 1000)

# plot the density & compute the kurtosis
plot(density(sample))
```

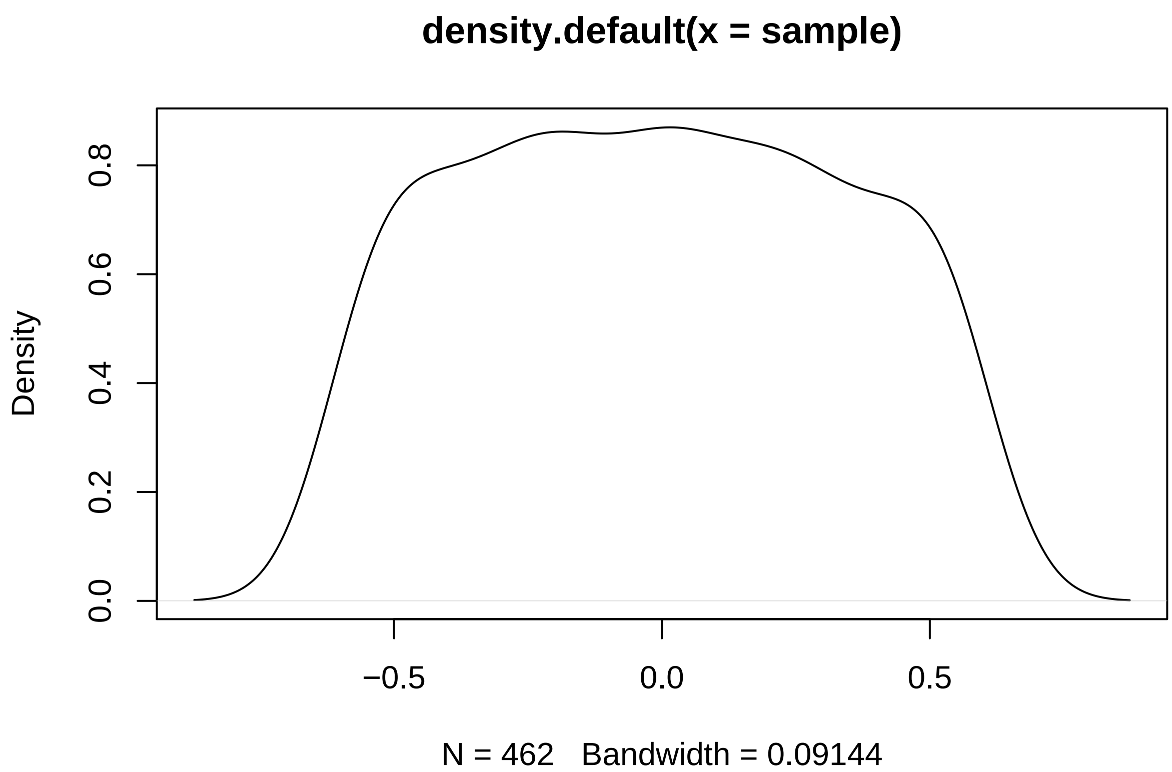



```
kurtosis(sample)
```

```
## [1] 3.233971
```

```
# now lets remove observations from the extremes in this distribution
# we thus intentionally alter the distribution to have less mass in its tails
sample <- sample[ sample > -0.6 & sample < 0.6]

# plot the distribution again and see how the tails of it (and thus the kurtosis has changed)
plot(density(sample))
```




```r
# re-calculate the kurtosis
kurtosis(sample)
```

```
## [1] 1.843965
```

```r
# as expected, the kurtosis has now a lower value
```

Implement the formulas for skewness and kurtosis in R

Skewness

```r
# own implementation
sum((sample-mean(sample))^3) / ((length(sample)-1) * sd(sample)^3)
```

```
## [1] 0.02736003
```

```r
# implementation in moments package
skewness(sample)
```

```
## [1] 0.02738969
```

Kurtosis

```r
# own implementation
sum((sample-mean(sample))^4) / ((length(sample)-1) * sd(sample)^4)
```

```
## [1] 1.839974
```

```r
# implementation in moments package
kurtosis(sample)
```

```
## [1] 1.843965
```