Simulation Using R In-built Functions

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

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
# A fun exercise to simulate the rolling of a fair die
# Sampling space for rolling a pair of fair dice
s = c(1:6)

# Corresponding probabilities
#p = c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
p = (1/6) * replicate(6, 1)

# The sampling process
nsimulation = 10
#sample(s, 1, replace = TRUE, prob = p) # Roll a dice with replacement
#sample(s, 2, replace = TRUE, prob = p) # Roll two dice
print("Rolling a dice 10 times:")
```

```
## [1] "Rolling a dice 10 times:"
```

```
replicate(nsimulation, sample(s, 1, replace = TRUE, prob = p))
```

```
##  [1] 4 1 1 4 2 4 6 2 5 4
```

```
print("Rolling 2 dice 10 times")
```

```
## [1] "Rolling 2 dice 10 times"
```

```
replicate(nsimulation, sample(s, 2, replace = TRUE, prob = p))
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,]    4    5    2    6    3    5    3    4    3     6
## [2,]    4    6    4    4    6    2    2    5    3     2
```

```
# Performing 100 simulations of rolling a dice
nsimulation = 100
simulatedData = replicate(nsimulation, sample(s, 1, replace = TRUE, prob = p))
str(simulatedData)
```

```
##  int [1:100] 2 2 2 4 4 3 4 1 2 3 ...
```

```r
print(simulatedData)
```

```
##   [1] 2 2 2 4 4 3 4 1 2 3 5 2 6 1 4 2 6 6 6 3 3 2 1 4 4 6 3 1 2 6 2 1 3 4 6 1 3
##  [38] 5 5 4 5 3 2 2 5 6 3 4 2 1 1 3 5 5 6 1 2 5 5 6 3 1 4 4 4 4 5 1 6 5 2 3 2 1
##  [75] 1 1 2 6 4 6 5 4 4 6 4 3 1 1 3 4 4 1 1 3 1 6 2 3 6 4
```

```r
nsimulation = 1e5
simulatedData = replicate(nsimulation, sample(s, 2, replace = TRUE, prob = p))

# Function to check if the sum of the rolls is at least 7
checkEvent = function(data){
  if (sum(data) >= 7) {
    return(1)
  } else{
    return(0)
  }
}

# Probability that the sum of the rolls is at least 7
#print(simulatedData)
#apply(simulatedData, 2, checkEvent)
mean(apply(simulatedData, 2, checkEvent))
```

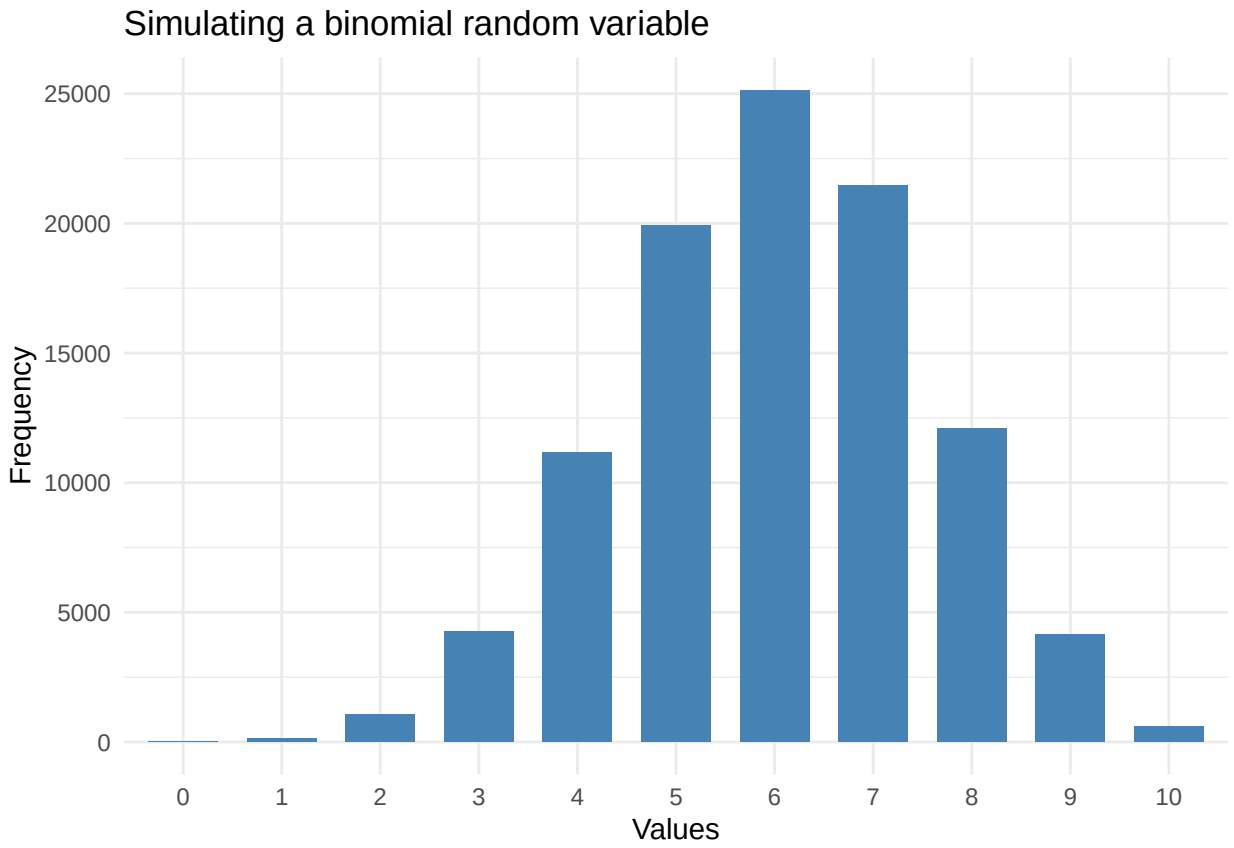
```
## [1] 0.58128
```

```r
# Function to check if the first roll is even
checkEvent1 = function(data){
  if (data[1] %% 2 == 0){
    return(1)
  } else{
    return(0)
  }
}
#apply(simulatedData, 2, checkEvent1)
mean(apply(simulatedData, 2, checkEvent1))
```

```
## [1] 0.49949
```

```r
# Probability that the first roll is even
```

```r
# Simulating a discrete random variable (n = 10, p = 0.6)
n = 10
p = 0.6
nsimulations = 1e5
simulatedData = rbinom(nsimulations, n, p)
df = as.data.frame(table(simulatedData))
colnames(df) = c('Value', 'Frequency')
#head(df)
p = ggplot(data = df) +
  geom_col(aes(x = Value, y = Frequency), width = 0.7, fill = 'steelblue') +
  ggtitle('Simulating a binomial random variable') +
  labs(x = 'Values', y = 'Frequency') +
  theme_minimal()
p
```




```
# Poissons Distribution
rpois(100, lambda = 10)
```

```
##   [1] 12 11  7 15 12  6 16 16  7 11 14 11 12 10 10 10  7 12  5 18  9 15 11 13 10
##  [26] 14  6 12  7  8  8 10  3  6  5 11  9 13 15 16 10  7  9 11 12 12  9  4 14 12
##  [51]  6  8 14 10  8 19 12  6 15 12  8  8  5  9 11 12 10  6 10 11  9 10  8 11 10
##  [76]  6 13  9 15  5  9 11  6  5 13  7  7 12 12 13 13  8  7 10 12 13  5  7  8 11
```

```
# Simulating a continuous random variable
mu = 170
sigma = 8
nsimulations = 1e5
simulatedData = rnorm(nsimulations, mean = mu, sd = sigma)
df = as.data.frame(simulatedData)
colnames(df) = 'Height'
head(df)
```

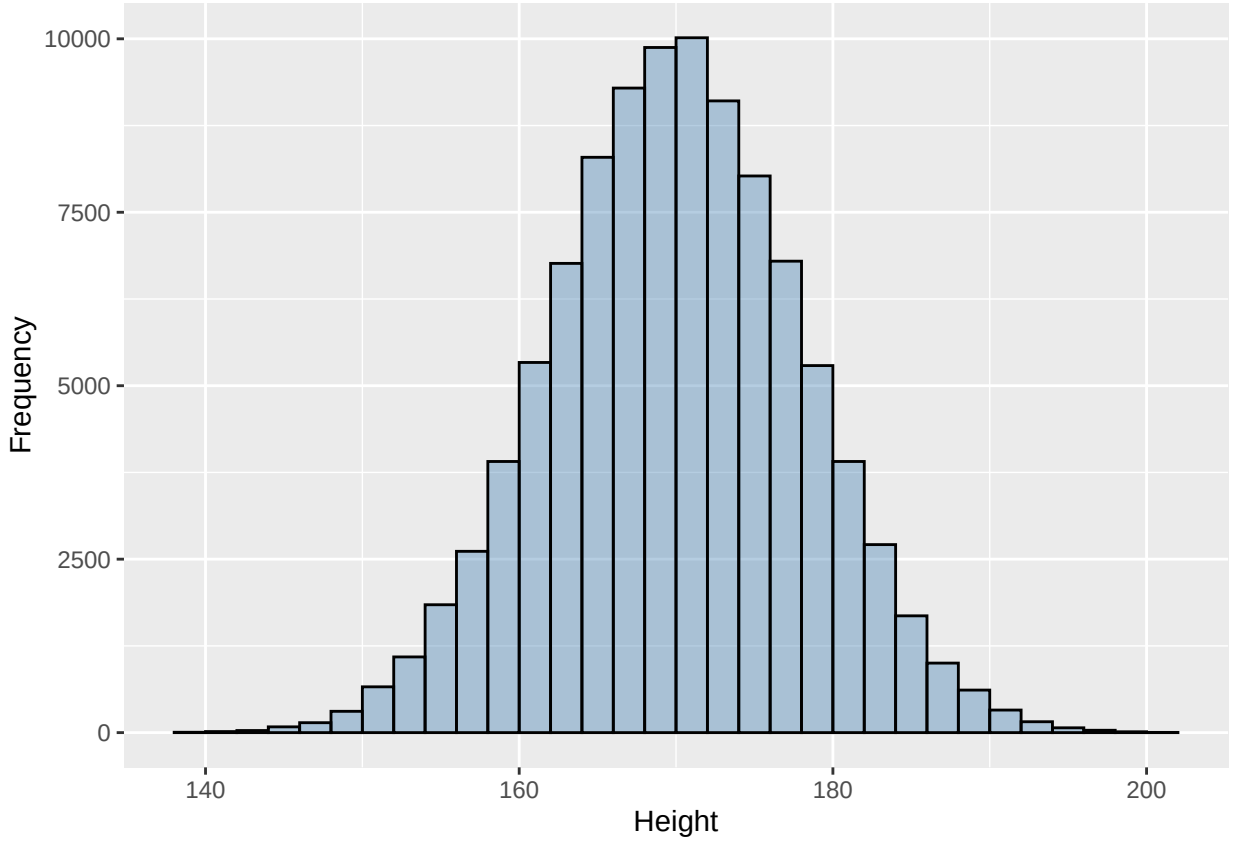
```
##     Height
## 1 164.9497
## 2 177.3149
## 3 170.1003
## 4 167.3985
## 5 172.9763
## 6 169.7939
```

```
mean((df['Height'] >= 170) & (df['Height'] <= 171))
```

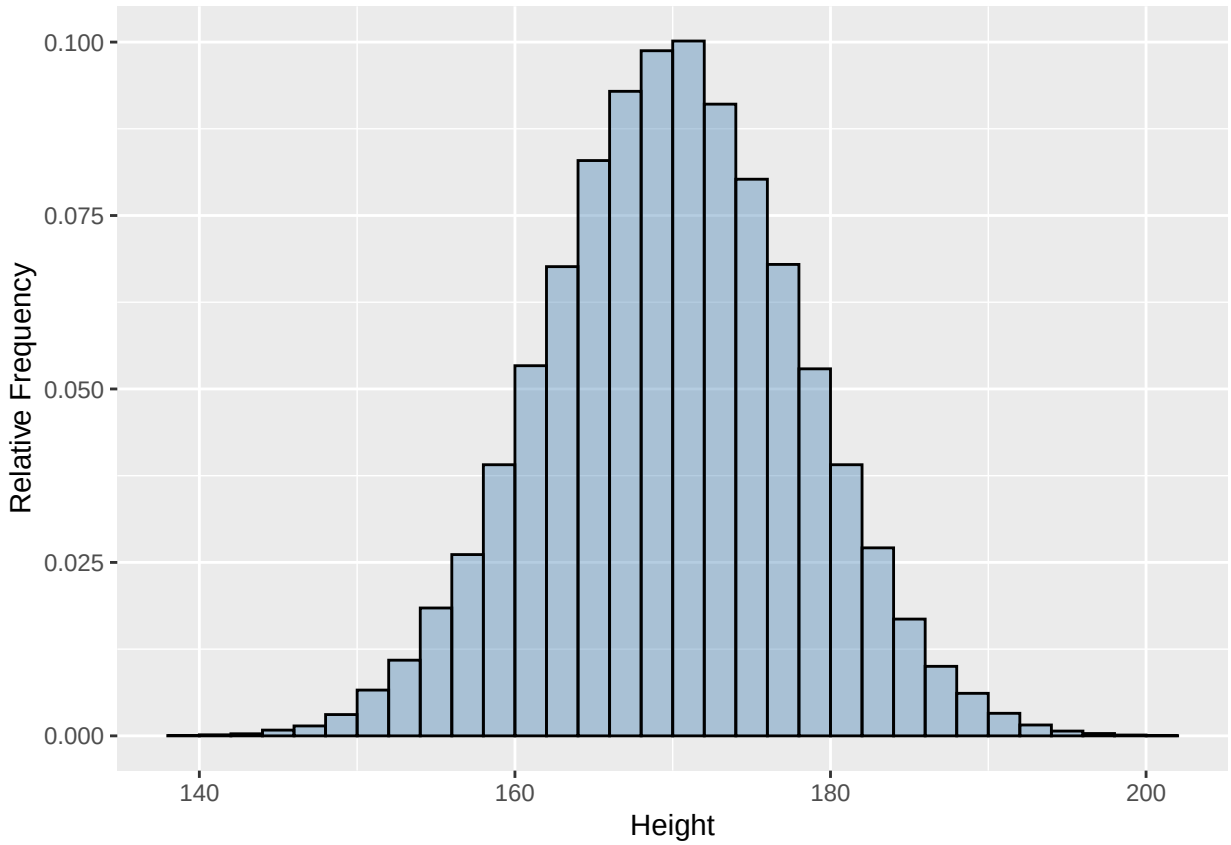
```
## [1] 0.05116
```

```r
delta = 2
p1 = ggplot(data = df) +
  geom_histogram(aes(x = Height, y = ..count..), breaks = seq(mu - 4*sigma, mu + 4*sigma, by = delta), 
  labs(x = 'Height', y = 'Frequency')
p1
```

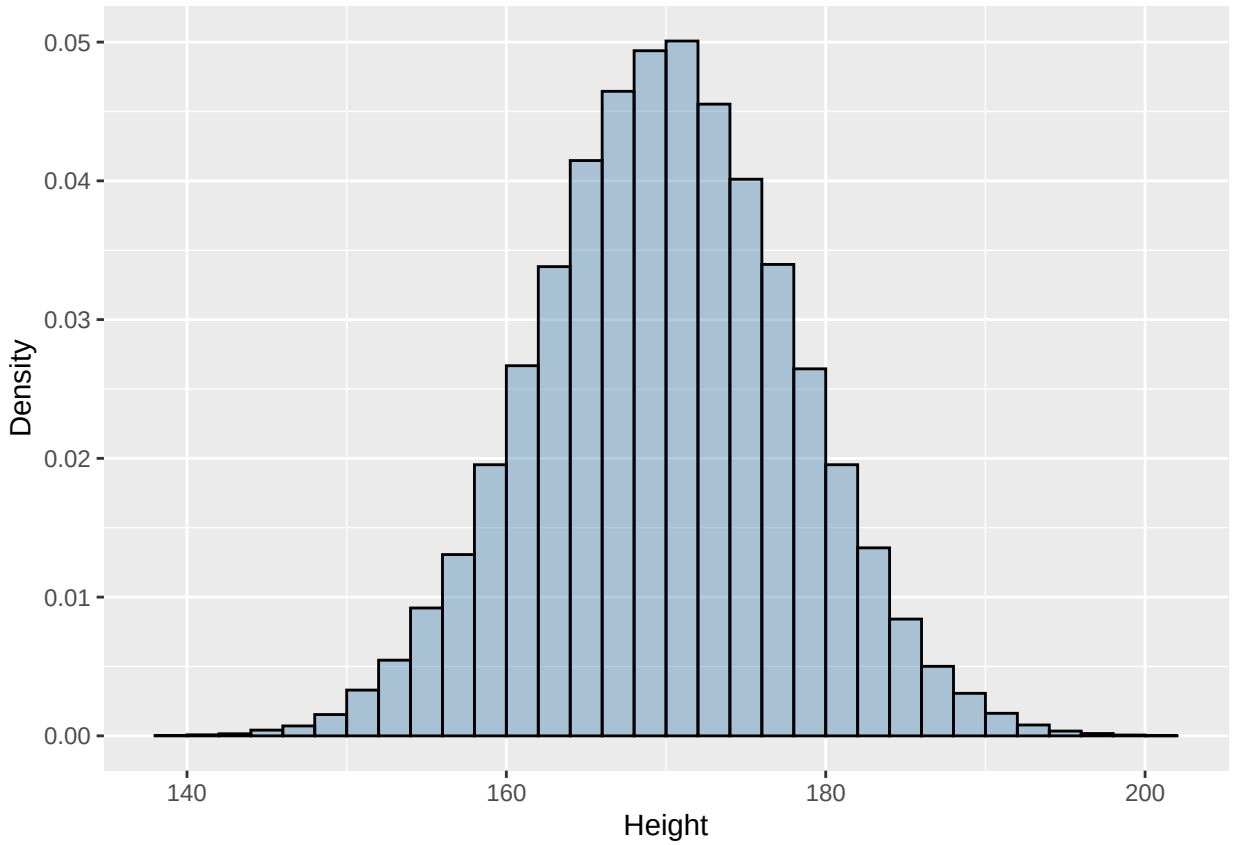
```
## Warning: The dot-dot notation (`..count..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(count)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```




```r
p2 = ggplot(data = df) +
  geom_histogram(aes(x = Height, y = ..count../sum(..count..)), breaks = seq(mu - 4*sigma, mu + 4*sigma
  labs(x = 'Height', y = 'Relative Frequency')
p2
```




```
p3 = ggplot(data = df) +
  geom_histogram(aes(x = Height, y = ..density..), breaks = seq(mu - 4*sigma, mu + 4*sigma, by = delta)
  labs(x = 'Height', y = 'Density')
p3
```


0.05
0.04
0.03
0.02
0.01
0.00
Density
140
160
180
200
Height