Predict 401 Quick Start Guide, Solutions

PREDICT 401

5 Working with Vectors

Exercises

1. Define the following vectors, “x” and “y.” Determine the answers to the questions (a) through (e) and check with R.

```
x <- seq(from = 1, to = 6)
y <- rep(x = 1:3, times = 2)

# (a) print(x) and print(y)
print(x)  # [1] 1 2 3 4 5 6
```

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

```
print(y)  # [1] 1 2 3 1 2 3
```

```
## [1] 1 2 3 1 2 3
```

```
# (b) combine the elements of y with x
c(x, y)  # [1] 1 2 3 4 5 6 1 2 3 1 2 3
```

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

```
# (c) find the length of c(x, y)
length(c(x, y))  # [1] 12
```

```
## [1] 12
```

```
# (d) sum the elements in c(x, y)
sum(c(x, y))  # [1] 33
```

```
## [1] 33
```

```
# (e) calculate x + y, x * y, x - 1, x ^ 2
x + y  # [1] 2 4 6 5 7 9
```

```
## [1] 2 4 6 5 7 9
```

```
x * y  # [1]  1  4  9  4 10 18
```

```
## [1]  1  4  9  4 10 18
```

```
x - 1  # [1] 0 1 2 3 4 5
```

```
## [1] 0 1 2 3 4 5
```

```
x ^ 2  # [1]  1  4  9 16 25 36
```

```
## [1]  1  4  9 16 25 36
```

2. Decide what the following expressions are and use R to check your answers.

```
# (a) seq(2, 9)
seq(from = 2, to = 9)  # [1] 2 3 4 5 6 7 8 9
```

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

```
# (b) seq(2, 9, length = 8)
seq(from = 2, to = 9, length = 8)  # [1] 2 3 4 5 6 7 8 9
```

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

```
# (c) seq(9, 2, -1)
seq(from = 9, to = 2, by = -1)  # [1] 9 8 7 6 5 4 3 2
```

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

```
# (d) rep(c(1, 2), 4)
rep(x = c(1, 2), times = 4)  # [1] 1 2 1 2 1 2 1 2
```

```
## [1] 1 2 1 2 1 2 1 2
```

```
# (e) rep(c(1, 2), c(4, 4))
rep(x = c(1, 2), times = c(4, 4))  # [1] 1 1 1 1 2 2 2 2
```

```
## [1] 1 1 1 1 2 2 2 2
```

```
# (f) rep(1:4, rep(3, 4))
rep(x = 1:4, times = rep(3, 4))  # [1] 1 1 1 2 2 2 3 3 3 4 4 4
```

```
##  [1] 1 1 1 2 2 2 3 3 3 4 4 4
```

3. Use the *rep()* function to define the following vectors in R.

```
# (a) 6, 6, 6, 6, 6, 6
rep(x = 6, times = 6)  # [1] 6 6 6 6 6 6
```

```
## [1] 6 6 6 6 6 6
```

```
# (b) 5, 8, 5, 8, 5, 8, 5, 8
rep(x = c(5, 8), times = 4)  # [1] 5 8 5 8 5 8 5 8
```

```
## [1] 5 8 5 8 5 8 5 8
```

```
# (c) 5, 5, 5, 5, 8, 8, 8, 8
rep(x = c(5, 8), times = c(4, 4))  # [1] 5 5 5 5 8 8 8 8
```

```
## [1] 5 5 5 5 8 8 8 8
```

6 Descriptive Statistics

Exercises

1. x <- c(8.75, 9.45, 4.35, 6.85, 9.45, 10.55, 7.75, 8.25, 10.55, 2.45, 15.75, 7.45) Determine the following and check with R.

```
x <- c(8.75, 9.45, 4.35, 6.85, 9.45, 10.55, 7.75, 8.25, 10.55, 2.45, 15.75, 7.45)

# (a) x[6] + x[7]
x[6] + x[7]  # [1] 18.3, ALTERNATIVELY, sum(x[6:7])
```

```
## [1] 18.3
```

```
# (b) x[c(5, 6, 7, 8)]
x[c(5, 6, 7, 8)]  # [1]  9.45 10.55  7.75  8.25
```

```
## [1]  9.45 10.55  7.75  8.25
```

```
# (c) x[5:8]
x[5:8]  # [1]  9.45 10.55  7.75  8.25
```

```
## [1]  9.45 10.55  7.75  8.25
```

```
# (d) x[c(1:4, 9:12)]
x[c(1:4, 9:12)]  # [1]  8.75  9.45  4.35  6.85 10.55  2.45 15.75  7.45
```

```
## [1]  8.75  9.45  4.35  6.85 10.55  2.45 15.75  7.45
```

```
# (e) The combination of the results from (c) and (d).
c(x[5:8], x[c(1:4, 9:12)])  # [1]  9.45 10.55  7.75  8.25  8.75  9.45  4.35  6.85 10.55  2.45
15.75  7.45
```

```
##  [1]  9.45 10.55  7.75  8.25  8.75  9.45  4.35  6.85 10.55  2.45 15.75
## [12]  7.45
```

2. x <- c(8.75, 9.45, 9.35, 9.85, 9.45, 10.55, 9.75, 8.25, 10.55, 9.45, 9.75, 8.45) The vector “x” contains birth weights in ounces of puppies from two different litters. Each litter had six puppies. The first six values are from the first litter, and the last six from the second litter. Produce a statistical summary of the birth weights for each litter.

```
x <- c(8.75, 9.45, 9.35, 9.85, 9.45, 10.55, 9.75, 8.25, 10.55, 9.45, 9.75, 8.45)

summary(x[1:6])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.750   9.375   9.450   9.567   9.750  10.550
```

```
summary(x[7:12])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   8.250   8.700   9.600   9.367   9.750  10.550
```

7 Logical Comparisons

Exercises

1. For y <- c(33, 44, 29, 16, 25, 45, 33, 19, 54, 22, 21, 49, 11, 24, 56) find the minimum and maximum of y and their location in the vector.

```
y <- c(33, 44, 29, 16, 25, 45, 33, 19, 54, 22, 21, 49, 11, 24, 56)

min(y)  # [1] 11, ALTERNATIVELY, y[which.min(y)]
```

```
## [1] 11
```

```
which.min(y)  # [1] 13
```

```
## [1] 13
```

```
max(y)  # [1] 56, ALTERNATIVELY, y[which.max(y)]
```

```
## [1] 56
```

```
which.max(y)  # [1] 15
```

```
## [1] 15
```

2. Find the median of y and use logicals to split y into two subsets. One subset will have all values in y strictly less than the median, and the other subset all values strictly greater than the median. Print the resulting subsets.

```
median(y)  # [1] 29
```

```
## [1] 29
```

```
less.than <- y[which(y < median(y))]
print(less.than)  # [1] 16 25 19 22 21 11 24
```

```
## [1] 16 25 19 22 21 11 24
```

```
greater.than <- y[which(y > median(y))]
print(greater.than)  # [1] 33 44 45 33 54 49 56
```

```
## [1] 33 44 45 33 54 49 56
```

8 Matrices

Exercises

1. Using the following code, create the matrices x and y. Perform the indicated calculations and check your answers in R.

```
x <- c(4, 3, 2, 1)
x <- matrix(x, nrow = 2)

y <- c(4, 9, 1, 16)
y <- matrix(y, nrow = 2)

# (a) 2 * x
2 * x
```

```
##      [,1] [,2]
## [1,]    8    4
## [2,]    6    2
```

```
# (b) sqrt(y)
sqrt(y)
```

```
##      [,1] [,2]
## [1,]    2    1
## [2,]    3    4
```

```
# (c) x * x
x * x
```

```
##      [,1] [,2]
## [1,]   16    4
## [2,]    9    1
```

```
# (d) x %*% x
x %*% x
```

```
##      [,1] [,2]
## [1,]   22   10
## [2,]   15    7
```

```
# (e) solve(y)
solve(y)
```

```
##            [,1]        [,2]
## [1,]  0.2909091 -0.01818182
## [2,] -0.1636364  0.07272727
```

```
# (f) round(solve(y) %*% y, digits = 1)
round(solve(y) %*% y, digits = 1)
```

```
##      [,1] [,2]
## [1,]    1    0
## [2,]    0    1
```

```
# (g) y[1, ]
y[1, ]
```

```
## [1] 4 1
```

```
# (h) y[, 2]
y[, 2]
```

```
## [1]  1 16
```

2. Define z <- c(2, 1). Calculate z %*% y and y %*% z. Check your answers in R.

```
z <- c(2, 1)

z %*% y
```

```
##      [,1] [,2]
## [1,]   17   18
```

```
y %*% z
```

```
##      [,1]
## [1,]    9
## [2,]   34
```

9 Accessing Example Datasets

Exercises

1. Use the dataset “women” and find the mean height and mean weight for individuals in the dataset. Type *help(women)* for a description of the variables available.

```
data(women)

mean(women$height)  # [1] 65
```

```
## [1] 65
```

```
mean(women$weight)  # [1] 136.7333
```

```
## [1] 136.7333
```

2. Use the *summary()* function to generate descriptive statistics for the dataset.

```
summary(women)
```

```
##      height         weight
##  Min.   :58.0   Min.   :115.0
##  1st Qu.:61.5   1st Qu.:124.5
##  Median :65.0   Median :135.0
##  Mean   :65.0   Mean   :136.7
##  3rd Qu.:68.5   3rd Qu.:148.0
##  Max.   :72.0   Max.   :164.0
```

10 The *apply()* Function

Exercises

1. Repeat the analyses shown above with *apply()* using the height and age in the “Loblolly” dataset. Repeat using the speed and distance data in the cars dataset. Compare your results to what is obtained using the *summary()* function.

```
data(Loblolly)
apply(Loblolly[, c("height", "age")], 2, mean)
```

```
##  height     age
## 32.3644 13.0000
```

```
summary(Loblolly)
```

```
##      height           age            Seed
##  Min.   : 3.46   Min.   : 3.0   329    : 6
##  1st Qu.:10.47   1st Qu.: 5.0   327    : 6
##  Median :34.00   Median :12.5   325    : 6
##  Mean   :32.36   Mean   :13.0   307    : 6
##  3rd Qu.:51.36   3rd Qu.:20.0   331    : 6
##  Max.   :64.10   Max.   :25.0   311    : 6
##                                 (Other):48
```

```
data(cars)
apply(cars, 2, mean)
```

```
## speed  dist
## 15.40 42.98
```

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

11 The *aggregate()* and *table()* Functions

Exercises

1. Use *aggregate()* to determine the median len for each combination of supp and dose.

```
data(ToothGrowth)
aggregate(len ~ supp + dose, data = ToothGrowth, FUN = median)
```

```
##   supp dose   len
## 1   OJ  0.5 12.25
## 2   VC  0.5  7.15
## 3   OJ  1.0 23.45
## 4   VC  1.0 16.50
## 5   OJ  2.0 25.95
## 6   VC  2.0 25.95
```

2. Use *with()* and *addmargins()* to reproduce the table of counts for the ToothGrowth data.

```
with(ToothGrowth, addmargins(table(ToothGrowth$supp, ToothGrowth$dose)))
```

```
##
##        0.5  1  2 Sum
##   OJ   10 10 10  30
##   VC   10 10 10  30
##   Sum  20 20 20  60
```

12 Loops

Exercises

1. Write a “for” loop to compute the value of a factorial for integers greater than zero. Execute the loop for an integer equal to 5, and print the factorials for 1, 2, 3, 4 and 5. Repeat the above, but with a “while” loop.

```
for (i in 1:5) {
  print(factorial(i))
}
```

```
## [1] 1
## [1] 2
## [1] 6
## [1] 24
## [1] 120
```

```
i <- 1
while (i <= 5) {
  print(factorial(i))
  i <- i + 1
}
```

```
## [1] 1
## [1] 2
## [1] 6
## [1] 24
## [1] 120
```

13 Writing Functions

Exercises

1. Write a simple function that computes the sample variance for a vector of numerical values. Use this function with *apply()* to compute the sample variance for the dimensions in the tree data.

```
data(trees)

SampVar <- function(x) {
  sum((x - mean(x))^2) / (length(x) - 1)
}
```

```
apply(trees, 2, SampVar)
```

```
##      Girth     Height     Volume
##   9.847914  40.600000 270.202796
```

14 Statistical Computation, Simulation and Random Sampling

Exercises

1. Suppose *X* has a normal distribution with mean 2 and variance 0.25. Denote by *f* and *F* the density and distribution functions.

```
# (a) Calculate the density function at 0.5, f(0.5) (use dnorm())
dnorm(0.4, mean = 2, sd = sqrt(0.25))  # [1] 0.004768176
```

```
## [1] 0.004768176
```

```
# (b) Calculate the distribution function value at 2.0, F(2.0) (use pnorm())
pnorm(2.0, mean = 2, sd = sqrt(0.25))  # [1] 0.5
```

```
## [1] 0.5
```

```
# (c) Calculate the 95th percentile (use qnorm())
qnorm(0.95, mean = 2, sd = sqrt(0.25))  # [1] 2.822427
```

```
## [1] 2.822427
```

```
# (d) Calculate the probability the X is between 1 and 3, Pr(1 <= X <= 3)
pnorm(3, mean = 2, sd = sqrt(0.25)) - pnorm(1, mean = 2, sd = sqrt(0.25))
```

```
## [1] 0.9544997
```

2. Repeat question 1 in the case that *X* has a *t*-distribution with 5 degrees of freedom

```
dt(0.5, df = 5)   # [1] 0.3279185
```

```
## [1] 0.3279185
```

```
pt(2.5, df = 5)   # [1] 0.972755
```

```
## [1] 0.972755
```

```
qt(0.95, df = 5)   # [1] 2.015048
```

```
## [1] 2.015048
```

```
pt(3, df = 5) - pt(1, df = 5)  # (d) [1] 0.1665591
```

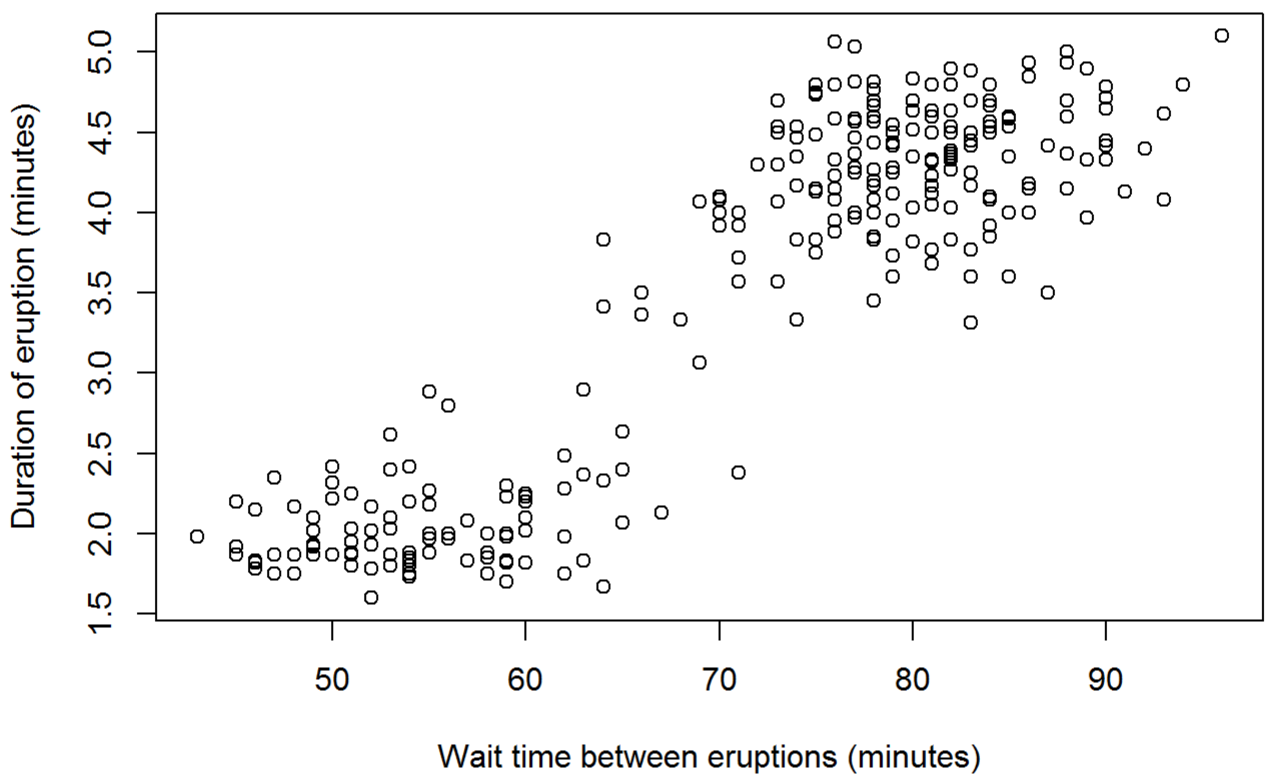
```
## [1] 0.1665591
```

15 Graphics

Exercises

1. Use the dataset “faithful” and create a scatterplot of the variables.

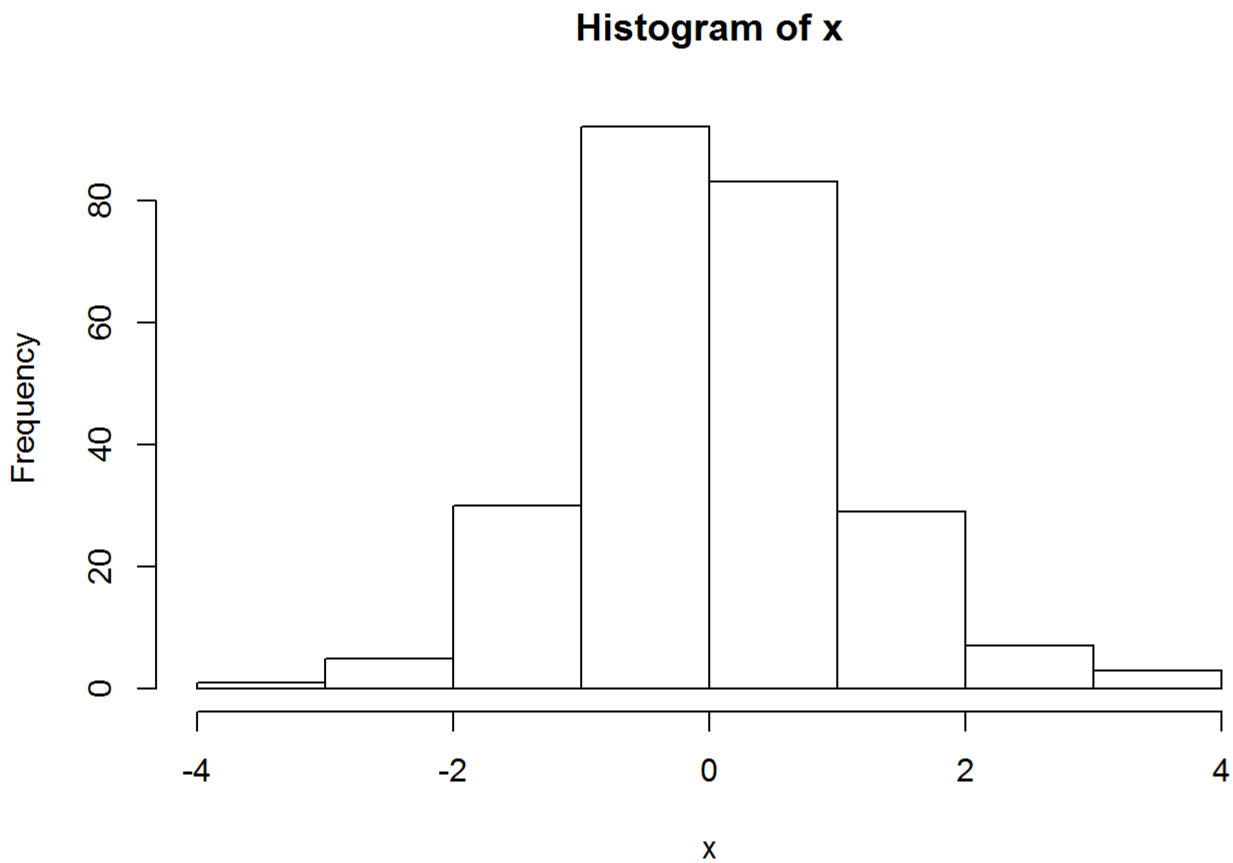
```
data(faithful)
plot(faithful$waiting, faithful$eruptions,
     xlab = "Wait time between eruptions (minutes)",
     ylab = "Duration of eruption (minutes)")
```



2. Use *x <- rnorm(250)* to generate 250 standard random values. Produce a histogram and compare to a stem-and-leaf plot of the data. The function *stem()* will produce the stem-and-leaf plot.

```
x <- rnorm(n = 250, mean = 0, sd = 1)

hist(x)
```




```
stem(x)
```

```
##
##   The decimal point is at the |
##
##   -3 | 2
##   -2 | 7
##   -2 | 220000
##   -1 | 977666
##   -1 | 444443333222222211100000
##   -0 | 9999988888887777777766666666665555555555
##   -0 | 44444444444444444333333333333322222222221111111100000
##    0 | 001111111111222233333333344444444444
##    0 | 555555555666666666667777777777777888889999999
##    1 | 00000111222223333444
##    1 | 55666667778
##    2 | 00012
##    2 | 57
##    3 | 112
```

16 Color Applications

Exercises

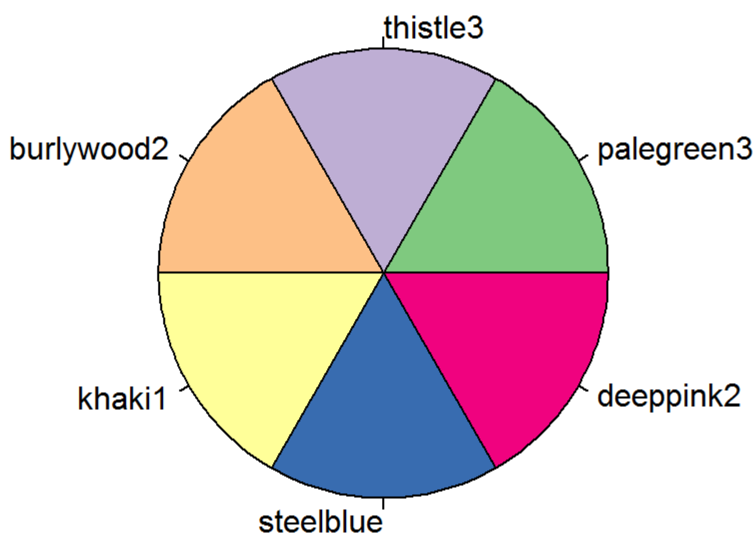
1. Use RColorBrewer and six colors from the Accent scheme to produce a pie chart with equal area slices. This will

require use of *brewer.pal()*.

```
library(RColorBrewer)
library(plotrix)
```

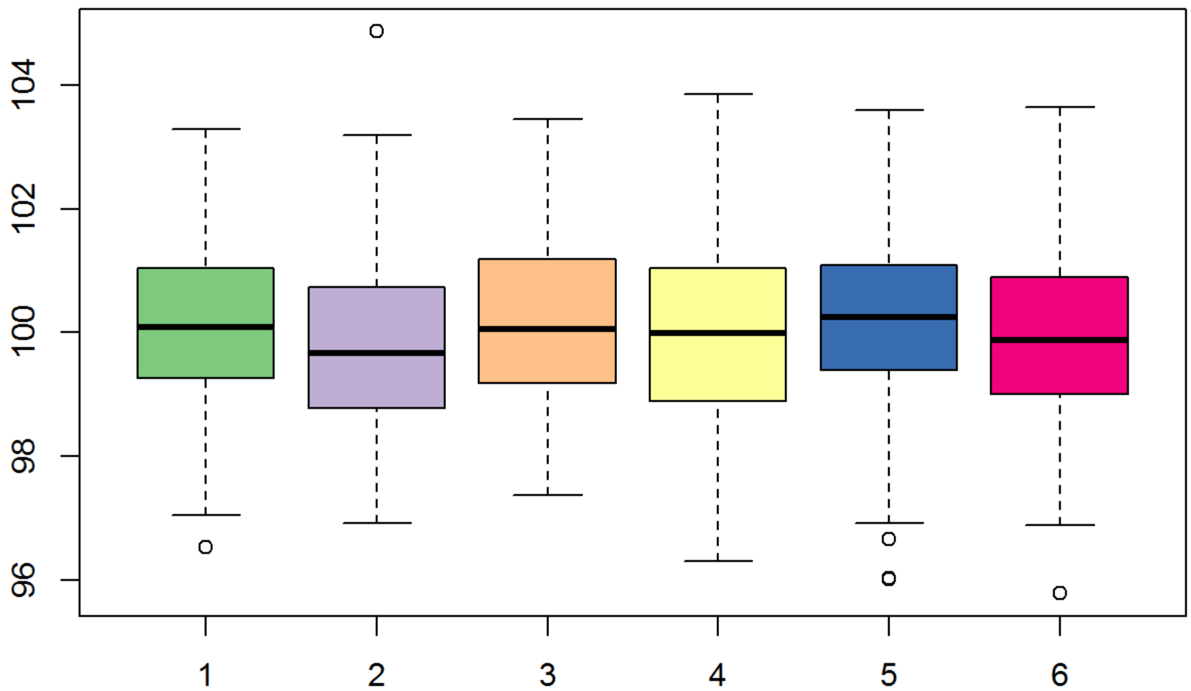
```
## Warning: package 'plotrix' was built under R version 3.2.5
```

```
pie(rep(1, 6), labels = sapply(brewer.pal(n = 6, name = "Accent"), color.id),
    col = brewer.pal(n = 6, name = "Accent"))
```



2. Use *set.seed(123)* and *rnorm(n = 100, mean = 100, sd = 1.5)* to generate six random samples. Use the color scheme from (1) above to produce six side-by-side boxplots.

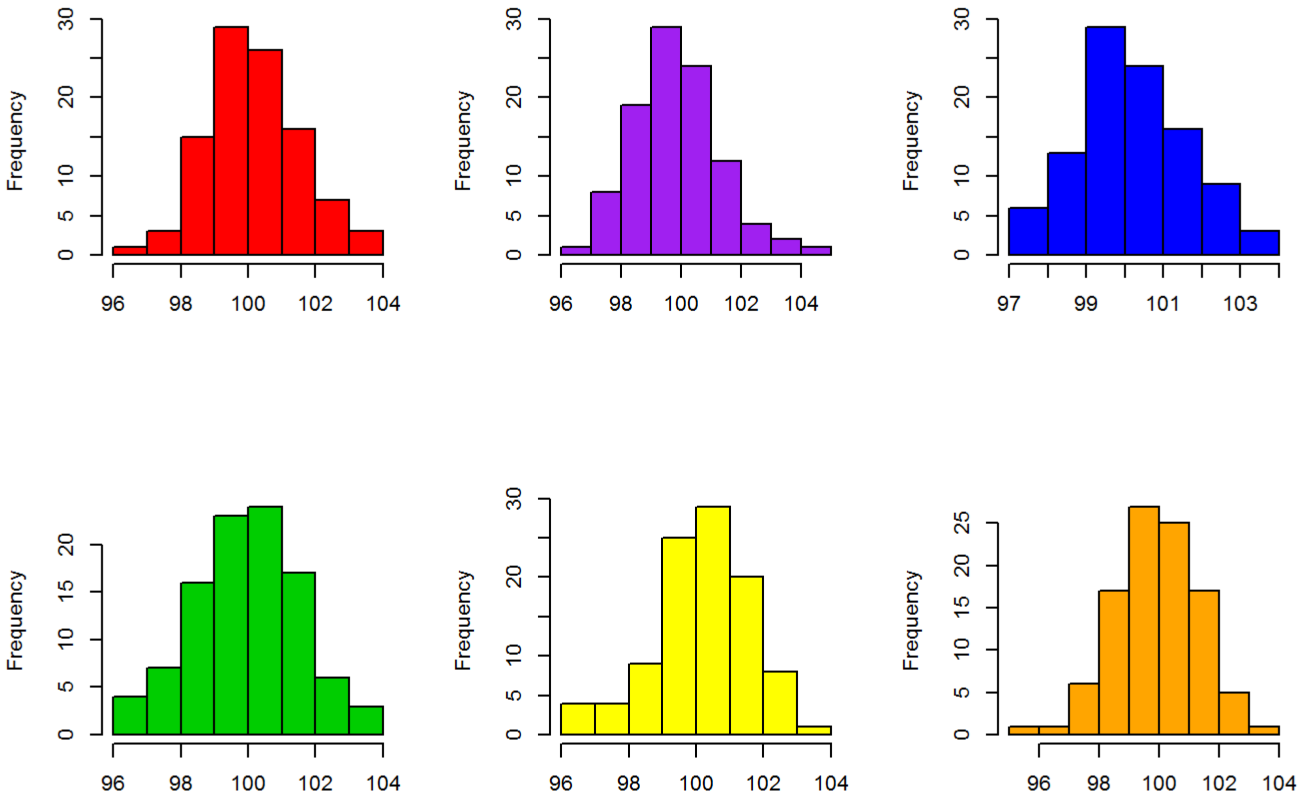
```
set.seed(123)
results <- replicate(6, rnorm(n = 100, mean = 100, sd = 1.5))
boxplot(results, col = brewer.pal(n = 6, name = "Accent"))
```



3. Produce six histograms using the samples from (2). Color each with a different color from the *palette(c("red", "purple", "blue", "green3", "yellow", "orange"))*.

```
palette(c("red", "purple", "blue", "green3", "yellow", "orange"))

par(mfrow = c(2, 3))
for (i in 1:6) {
  hist(results[, i],
       col = palette()[i],
       xlab = "",
       main = "")
}
```




```
par(mfrow = c(1, 1))
```

4. Use *col2rgb()* and determine the *rgb()* representation for the six colors in the palette used in (3) above. Convert the rgb codes into hex codes.

```
color.mat <- col2rgb(palette())

colorID <- function(red, green, blue, max) {
    object <- rgb(red, green, blue, maxColorValue = max)
    return(object)
}

color.list <- c(NULL)

# Build color list
for (k in 1:6) {
    item <- colorID(color.mat[1, k], color.mat[2, k], color.mat[3, k], 255)
    color.list <- c(color.list, item)
}

color.list # [1] "#FF0000" "#A020F0" "#0000FF" "#00CD00" "#FFFF00" "#FFA500"
```

```
## [1] "#FF0000" "#A020F0" "#0000FF" "#00CD00" "#FFFF00" "#FFA500"
```