Live session assignment 05

Samira Zarandioon

2/19/2017

Question 1

1. Standard deviation of values less than 45:

```
testdata <- c(45.4,44.2,36.8,35.1,39.0,60.0,47.4,41.1,45.8,35.6)
sd(testdata[testdata<45])
```

```
## [1] 3.52855
```

2. Number of vectors entries that are less than 45:

```
sum(testdata<45)
```

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

3. Number of entries between 40 and 55:

```
sum(testdata>40 & testdata<55)
```

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

4. Proportion of data vector exceeding 40:

```
n1 <- sum(testdata>40)
l <- length(testdata)
n1/l
```

```
## [1] 0.6
```

Question 2:

```
cigsales_df <- read.table("./cigsales.txt",header=TRUE)
```

1. States that have over a 15% African American population:

```
cigsales_df[cigsales_df$black>15,c("state")]
```

```
##  [1] AL AR DC FL GA LA MD MS NC SC TN VA
## 51 Levels: AK AL AR AZ CA CO CT DC DE FL GA HI ID IL IN IO KA KY LA ... WY
```

or:

```
subset(cigsales_df, black>15, select = c("state"))
```

```
##    state
## 1     AL
## 4     AR
## 9     DC
## 10    FL
## 11    GA
## 19    LA
```

```
## 21    MD
## 25    MS
## 34    NC
## 41    SC
## 43    TN
## 47    VA
```

2. Price vector:

```
price.vec <- cigsales_df[,c("price")]
price.vec
```

```
##  [1] 42.7 41.8 38.5 38.8 39.7 31.1 45.5 41.3 32.6 43.8 35.8 36.7 33.6 41.4
## [15] 32.2 38.5 38.9 30.1 39.3 38.8 34.2 41.0 39.2 40.1 37.5 36.8 34.7 34.7
## [29] 44.0 34.1 41.7 41.7 41.7 29.4 38.9 38.1 39.8 29.0 44.7 40.2 34.3 38.5
## [43] 41.6 42.0 36.6 39.5 30.2 40.3 41.6 40.2 34.4
```

3. Poor vs. rich:

```
income_median <- median(cigsales_df$income)
poor <- cigsales_df[cigsales_df$income < income_median,c("income")]
poor
```

```
##  [1] 2948 3665 2878 3738 3354 3290 3112 3090 3302 2626 3500 3737 3077 3252
## [15] 3086 3387 3719 2990 3123 3119 3606 3227 3468 3712 3061
```

```
rich <- cigsales_df[cigsales_df$income >= income_median,c("income")]
rich
```

```
##  [1] 4644 4493 3855 4917 4524 5079 4623 4507 3772 3751 3853 4309 4340 4180
## [15] 3859 3781 3789 4563 4701 4712 4020 3971 3959 4053 3812 3815
```

Question 3

1. Example 1: Function *trim* gets a vector as well as lower and upper bound and as output returns a vector containing the elements that are between the lower and upper bound. See the following execution as an example.

```
trim <-function(x, lower=0.0, upper=1.0){
indicies <- x >= lower & x<= upper
return (x[indicies])
}
data <- c(3,6,2,1,8,10)
trim(data,3,8)
```

```
## [1] 3 6 8
```

2. Example 2: This function operates exactly as the previous function. The only differance is that its definition is more consise.

```
trim <- function(x, lower=0.0, upper=1.0) x[x>=lower & x<=upper]
data <- c(3,6,2,1,8,10)
trim(data,3,8)
```

```
## [1] 3 6 8
```

3. Example 3: Function *sumdice* simulates throughing a fair six-sided die for n times and returns the sum of the dots that comes up. The following illustrates an application of the function:

```
sumdice <- function(n) {
k<-sample(1:6, size=n,replace= TRUE)
return(sum(k))
}
sumdice(5)
```

```
## [1] 20
```

Question 4:

We expect to get (1+6)/2 (ie. 3.5) dots. To get a reliable estimate of number of dice we can call *sumdice* function with a sufficiently large n (eg. 100000) and then divided by n:

```
sumdice(100000)/100000
```

```
## [1] 3.50189
```

Question 5:

There are two variable x: one x is defined outside the function and another one is defined inside the function definition. The scope of x inside the function is limited within the function, while the outside x is global. The internal x hights the external x, therefore the assignment inside function does not impact the external x. The variable y is the input parameters of the function and similar to the internal x it's scope is limited to the function definition. The following code execution illustrates this:

```
x<-42
fred <- function(y) {
  x <- y
  return (y+x)
}
fred(13)
```

```
## [1] 26
```

```
x
```

```
## [1] 42
```

Question 6:

```
diff <- function(v) abs(mean(v) - median(v))
diff(price.vec)
```

```
## [1] 0.8254902
```

Question 7:

It prints numbers from 4 to 23. See the following for a sample execution:

```r
for (i in 1:20) {
  print(i+3)
}
```

```
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
```

Question 8:

We can compute the probablities by drawing large number of samples from the normal distributions and dividing number of samples that have the specific condition by the total number of samples. The following lagic computes the requested probablities:

1. X <= 66 :

```r
n <- 10000000
v <- rnorm(n, mean=60, sd=60)
length(v[v<=66])/n
```

```
## [1] 0.5401572
```

2. 50 < X < 60 :

```r
n <- 10000000
v <- rnorm(n, mean=60, sd=60)
length(v[v<60 & v>5])/n
```

```
## [1] 0.3204218
```

3. X > 68 :

```r
n <- 10000000
v <- rnorm(n, mean=60, sd=60)
length(v[v>68])/n
```

```
## [1] 0.4470236
```

Question 9:

The HTML file of the live session assignment 04 is included in the file called *assignment_4.html*.