Homework 2

Angela Zhang

10/9/2017

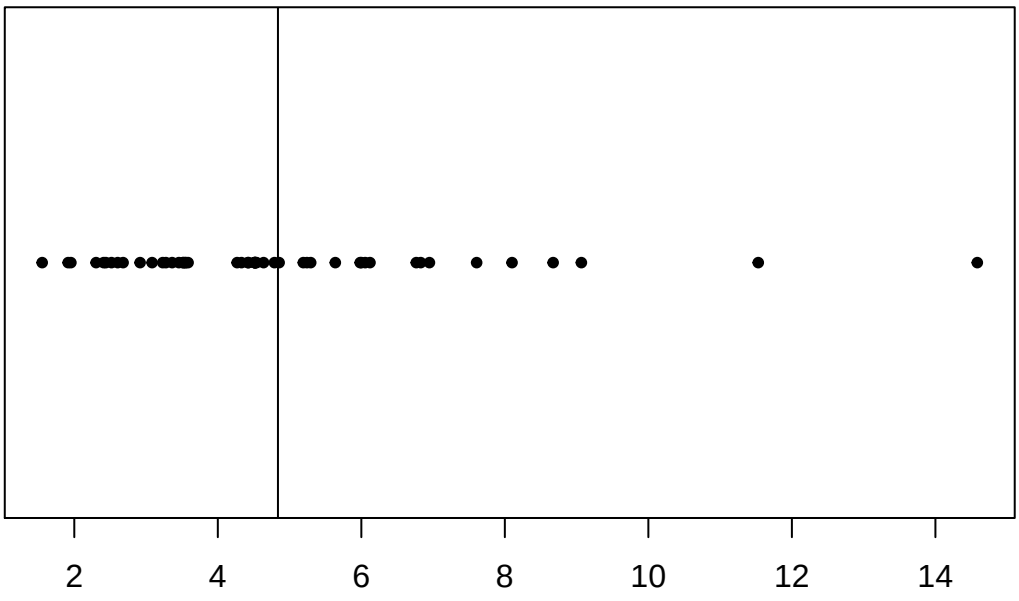
Writing your own function

I named my function, myFunction, that takes the options, "data", "na.rm", and "abline". The "if" loop specifies whether or not the abline shows the median or the mean.

```
myFunction = function(data,na.rm=T,abline="mean"){
  mean_data = mean(data, na.rm = na.rm)
  median_data = median(data, na.rm = na.rm)
  stripchart(data,pch =20)
  if (abline == "mean"){
    abline(v = mean_data)
  }
  else{
    abline(v = median_data)
  }
  central = c(mean_data, median_data, as.numeric(sum(is.na(data))))
  names(central) = c("Mean","Median","Number of NAs")
  print(central)
}


set.seed(1234)
my_data <- c(rgamma(50, shape = 5, rate = 1), NA)

myFunction(my_data, na.rm=T)
```




```
##          Mean        Median Number of NAs
##      4.838115      4.512631      1.000000
```

Looping with functional programming

(a)

```
mean_sd = function(x, ...){
  c(mean(x, ...), sd = sd(x, ...))
}

apply(airquality, 2, function(x){by(x, list(toohot = airquality$Temp > 85), mean_sd)})
```

```
## $Ozone
## toohot: FALSE
##    sd
## NA NA
## ------------------------------------------------------------
## toohot: TRUE
##    sd
## NA NA
##
## $Solar.R
## toohot: FALSE
##    sd
## NA NA
## ------------------------------------------------------------
## toohot: TRUE
##    sd
## NA NA
##
## $Wind
## toohot: FALSE
##                   sd
## 10.594958  3.408062
## ------------------------------------------------------------
## toohot: TRUE
##                sd
## 7.726471 3.007094
##
## $Temp
## toohot: FALSE
##                   sd
## 74.495798  7.779671
## ------------------------------------------------------------
## toohot: TRUE
##                   sd
## 89.735294  3.184325
##
## $Month
## toohot: FALSE
##                sd
## 6.831933 1.491895
## ------------------------------------------------------------
## toohot: TRUE
##                   sd
## 7.5588235 0.9274003
##
```

```
## $Day
## toohot: FALSE
##                   sd
## 16.302521  8.578209
## ------------------------------------------------------------
## toohot: TRUE
##                   sd
## 14.058824  9.735724
```

The results produce the same numbers but looks much messier than subsetting the list first by temperature and then finidng the mean and standard deviation of the columns.

(b) I will use the dataset Beaver1

`lapply` returns a list of values. It takes a dataset and applies a function to all columns of the dataset. For example, I took the mean of all the variables in the beaver1 dataset and I got the means for day, time, temp, and activity.

```
lapply(beaver1, mean)
```

```
## $day
## [1] 346.2018
##
## $time
## [1] 1312.018
##
## $temp
## [1] 36.86219
##
## $activ
## [1] 0.05263158
```

The `mapply` function gives us a way to use non-vectorized functions. For example, if you want to create a matrix with the same rows, instead of

```
matrix(c(rep(1, 3), rep(2, 3), rep(3, 3)),nrow=3,ncol=3)
```

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

We can use `mapply` instead. This takes the "rep" function with the vector 1:3, and replicates it three times

```
mapply(rep,1:3,3)
```

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

Bootstrapping

I entered the response as a vector containing the 5 responses. From Lecture 2 notes, I sampled with replacement from the "response" variable 10,000 times to get 10,000 variables. With this vector, I applied the 'var' function to get the variance.

```r
response = c(1,5,8,3,7)

many_medians = replicate(10000, median(sample(response, size = 5, replace = T)))
var(many_medians)
```

```
## [1] 3.532823
```