Lab2

Ankita Shankhdhar

Due 25 September 2015

```
library(MASS)
```

```
## Warning: package 'MASS' was built under R version 3.1.3
```

```
data(Cars93)
View(Cars93)
```

Part 1: Data Frames

Question 1:

Cars93 has 93 rows and and 27 columns or in other words it has 93 observcations and 27 variables. I could only tell the number of variables (columns) using the summary function. I could not tell the number of observations (rows). I used the dim function to evaluate the rows and the columns

```
summary(Cars93)
```

```
##     Manufacturer     Model         Type      Min.Price         Price      
##  Chevrolet: 8    100    : 1   Compact:16   Min.   : 6.70   Min.   : 7.40  
##  Ford     : 8    190E   : 1   Large  :11   1st Qu.:10.80   1st Qu.:12.20  
##  Dodge    : 6    240    : 1   Midsize:22   Median :14.70   Median :17.70  
##  Mazda    : 5    300E   : 1   Small  :21   Mean   :17.13   Mean   :19.51  
##  Pontiac  : 5    323    : 1   Sporty :14   3rd Qu.:20.30   3rd Qu.:23.30  
##  Buick    : 4    535i   : 1   Van    : 9   Max.   :45.40   Max.   :61.90  
##  (Other)  :57    (Other):87                                               
##    Max.Price       MPG.city      MPG.highway                  AirBags  
##  Min.   : 7.9   Min.   :15.00   Min.   :20.00   Driver & Passenger:16  
##  1st Qu.:14.7   1st Qu.:18.00   1st Qu.:26.00   Driver only       :43  
##  Median :19.6   Median :21.00   Median :28.00   None              :34  
##  Mean   :21.9   Mean   :22.37   Mean   :29.09                          
##  3rd Qu.:25.3   3rd Qu.:25.00   3rd Qu.:31.00                          
##  Max.   :80.0   Max.   :46.00   Max.   :50.00                          
##                                                                        
##  DriveTrain  Cylinders    EngineSize      Horsepower         RPM      
##  4WD  :10   3      : 3   Min.   :1.000   Min.   : 55.0   Min.   :3800  
##  Front:67   4      :49   1st Qu.:1.800   1st Qu.:103.0   1st Qu.:4800  
##  Rear :16   5      : 2   Median :2.400   Median :140.0   Median :5200  
##             6      :31   Mean   :2.668   Mean   :143.8   Mean   :5281  
##             8      : 7   3rd Qu.:3.300   3rd Qu.:170.0   3rd Qu.:5750  
##             rotary: 1    Max.   :5.700   Max.   :300.0   Max.   :6500  
##                                                                       
##   Rev.per.mile  Man.trans.avail Fuel.tank.capacity   Passengers   
##  Min.   :1320   No :32          Min.   : 9.20      Min.   :2.000  
##  1st Qu.:1985   Yes:61          1st Qu.:14.50      1st Qu.:4.000  
##  Median :2340                   Median :16.40      Median :5.000  
##  Mean   :2332                   Mean   :16.66      Mean   :5.086  
##  3rd Qu.:2565                   3rd Qu.:18.80      3rd Qu.:6.000
##  Max.   :3755                   Max.   :27.00      Max.   :8.000
##
##      Length        Wheelbase         Width        Turn.circle
##  Min.   :141.0   Min.   : 90.0   Min.   :60.00   Min.   :32.00
##  1st Qu.:174.0   1st Qu.: 98.0   1st Qu.:67.00   1st Qu.:37.00
##  Median :183.0   Median :103.0   Median :69.00   Median :39.00
##  Mean   :183.2   Mean   :103.9   Mean   :69.38   Mean   :38.96
##  3rd Qu.:192.0   3rd Qu.:110.0   3rd Qu.:72.00   3rd Qu.:41.00
##  Max.   :219.0   Max.   :119.0   Max.   :78.00   Max.   :45.00
##
##  Rear.seat.room   Luggage.room       Weight         Origin
##  Min.   :19.00   Min.   : 6.00   Min.   :1695   USA    :48
##  1st Qu.:26.00   1st Qu.:12.00   1st Qu.:2620   non-USA:45
##  Median :27.50   Median :14.00   Median :3040
##  Mean   :27.83   Mean   :13.89   Mean   :3073
##  3rd Qu.:30.00   3rd Qu.:15.00   3rd Qu.:3525
##  Max.   :36.00   Max.   :22.00   Max.   :4105
##  NA's   :2       NA's   :11
##             Make
##  Acura Integra: 1
##  Acura Legend : 1
##  Audi 100     : 1
##  Audi 90      : 1
##  BMW 535i     : 1
##  Buick Century: 1
##  (Other)      :87
```

```r
dim(Cars93)
```

```
## [1] 93 27
```

Question 2:

The max price of a car with a rear-wheel drive train is:

```r
rear <-Cars93[Cars93$DriveTrain=="Rear",]
mean(rear$Price)
```

```
## [1] 28.95
```

Question 3:

The minimum horsepower of all cars with capacity for 7 passengers is:

```r
cap.7 <-Cars93[Cars93$Passengers=="7",]
min(cap.7$Horsepower)
```

```
## [1] 109
```

The minimum horsepower of all cars with capacity of at least 6 passengers is:

```
cap.6 <-Cars93[Cars93$Passengers>="6",]
min(cap.6$Horsepower)
```

```
## [1] 100
```

Question 4:

To compute the distance travellable we need miles per gallon that it can travel on the highway and the fuel tank capacity of the car itself. Looking at these two we know that the miles it can travel in a gallon and multiplying that with the amount og fuel it can hold in gallons would give you the amount it can travel overall.

```
distance.travellable <- (Cars93$MPG.highway)*(Cars93$Fuel.tank.capacity)
max(distance.travellable)
```

```
## [1] 633
```

```
min(distance.travellable)
```

```
## [1] 288.6
```

```
median(distance.travellable)
```

```
## [1] 470.4
```

Part II - Reproducibility and Functions

```
factory.function <- function (cars.output=1, trucks.output=1) {
  factory <- matrix(c(40,1,60,3),nrow=2,
    dimnames=list(c("labor","steel"),c("cars","trucks")))
  available <- c(1600,70); names(available) <- rownames(factory)
  slack <- c(8,1); names(slack) <- rownames(factory)
  output <- c(cars.output, trucks.output); names(output) <- colnames(factory)

  passes <- 0 # How many times have we  been around the loop?
  repeat {
     passes <- passes + 1
     needed <- factory %*% output # What do we need for that output level?
     # If we're not using too much, and are within the slack, we're done
     if (all(needed <= available) &&
         all((available - needed) <= slack)) {
       break()
     }
     # If we're using too much of everything, cut back by 10%
     if (all(needed > available)) {
       output <- output * 0.9
       next()
     }
     # If we're using too little of everything, increase by 10%
     if (all(needed < available)) {
       output <- output * 1.1
```

```
        next()
      }
      # If we're using too much of some resources but not others, randomly
      # tweak the plan by up to 10%
       # runif == Random number, UNIFormly distributed, not "run if"
      output <- output * (1+runif(length(output),min=-0.1,max=0.1))
  }

  return(output)
}
```

Question 5:

If we want to optimize a function then starting with any initial guess we should be able to converge.

```
factory.function()
```

```
##      cars    trucks
##  9.956428 20.006219
```

Question:6

These values are around the same for cars and trucks. The output value of cars is approximately 10 and the output value of trucks is approximately 20.

```
factory.function(0.005,0.005)
```

```
##     cars   trucks
## 10.09951 19.89895
```

```
factory.function(0.05,0.05)
```

```
##     cars   trucks
## 10.18076 19.80749
```

```
factory.function(0.5,0.5)
```

```
##      cars    trucks
##  9.829629 20.045719
```

```
factory.function(5,5)
```

```
##     cars   trucks
## 10.17877 19.86869
```

```
factory.function(50,50)
```

```
##     cars   trucks
## 10.29824 19.70946
```

Question 7:

```
factory.function2 <- function (cars.output=1, trucks.output=1) {
  factory <- matrix(c(40,1,60,3),nrow=2,
    dimnames=list(c("labor","steel"),c("cars","trucks")))
  available <- c(1600,70); names(available) <- rownames(factory)
  slack <- c(8,1); names(slack) <- rownames(factory)
  output <- c(cars.output, trucks.output); names(output) <- colnames(factory)

  passes <- 0 # How many times have we  been around the loop?
  repeat {
     passes <- passes + 1
     needed <- factory %*% output # What do we need for that output level?
     # If we're not using too much, and are within the slack, we're done
     if (all(needed <= available) &&
         all((available - needed) <= slack)) {
       break()
     }
     # If we're using too much of everything, cut back by 10%
     if (all(needed > available)) {
       output <- output * 0.9
       next()
     }
     # If we're using too little of everything, increase by 10%
     if (all(needed < available)) {
       output <- output * 1.1
       next()
     }
     # If we're using too much of some resources but not others, randomly
     # tweak the plan by up to 10%
      # runif == Random number, UNIFormly distributed, not "run if"
     output <- output * (1+runif(length(output),min=-0.1,max=0.1))
  }

  return(c(output,"Passes"=passes))
}
```

```
factory.function2()
```

```
##        cars      trucks      Passes
##     9.79441    20.01957 35958.00000
```

```
factory.function2(0.005,0.005)
```

```
##       cars     trucks     Passes
##   10.31476   19.77114 5945.00000
```

```
factory.function2(0.05,0.05)
```

```
##       cars     trucks     Passes
##   9.952685  19.966321 141.000000
```

```
factory.function2(0.5,0.5)
```

```
##      cars    trucks    Passes
##  10.00741  19.91487 569.00000
```

```
factory.function2(5,5)
```

```
##      cars    trucks    Passes
##  10.28065  19.79202 494.00000
```

```
factory.function2(50,50)
```

```
##       cars     trucks     Passes
##   10.15912   19.80803 2037.00000
```

We notice that the number of passes changes everytime. The number of iterations changes everytime due to the randomness.

Question 8:

```
factory.function3 <- function (cars.output=1, trucks.output=1) {
  factory <- matrix(c(40,1,60,3),nrow=2,
    dimnames=list(c("labor","steel"),c("cars","trucks")))
  available <- c(1600,70); names(available) <- rownames(factory)
  slack <- c(8,1); names(slack) <- rownames(factory)
  output <- c(cars.output, trucks.output); names(output) <- colnames(factory)

  passes <- 0 # How many times have we  been around the loop?
  repeat {
     passes <- passes + 1
     needed <- factory %*% output # What do we need for that output level?
     # If we're not using too much, and are within the slack, we're done
     if (all(needed <= available) &&
         all((available - needed) <= slack)) {
       break()
     }
     # If we're using too much of everything, cut back by 10%
     if (all(needed > available)) {
       output <- output * 0.9
       next()
     }
     # If we're using too little of everything, increase by 10%
     if (all(needed < available)) {
       output <- output * 1.1
       next()
     }
     # If we're using too much of some resources but not others, randomly
     # tweak the plan by up to 10%
      # runif == Random number, UNIFormly distributed, not "run if"
     output <- output * (1+runif(length(output),min=-0.1,max=0.1))
  }
  return(c(output,"Passes"=passes,"Needed"=needed,"Slack"=slack,"Available"=available))
}

factory.function3(30,20)
```

```
##            cars          trucks          Passes         Needed1
##        10.10538        19.88474      1329.00000      1597.29967
##         Needed2     Slack.labor     Slack.steel Available.labor
##        69.75960         8.00000         1.00000      1600.00000
## Available.steel
##        70.00000
```

We clearly want more than what we get from our resources. We want to make 30 cars and 20 trucks. So this means that our plan is not within our budget and so not within our slack.