Stat597A: Lab 2, Sept 18 2015

Thananya Saksuriyongse

Part I – Data Frames

R includes a number of pre-specified data objects as part of its default installation. We will load and manipulate one of these, a data frame of 93 cars with model year 1993. Begin by ensuring that you can load this data with the commands

```r
library(MASS)
data(Cars93)
```

Begin by examining the data frame with the command `View(Cars93)` to understand the underlying object. You will need to use functions and other commands to extract elements for this assignment.

1. Obtain a `summary()` of the full data structure. Can you tell from this how many rows are in the data? If so, say how; if not, use another method to obtain the number of rows.

```r
summary(Cars93)      #may know the column but not row
```

```r
dim(Cars93)          # number of row and column of Cars93
```

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

```r
nrow(Cars93)         # number of row of Cars93
```

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

2. What is the mean price of a car with a rear-wheel drive train?

```r
index <-  Cars93$DriveTrain == "Rear"
rear_wheel <- Cars93[index, ]          #get all info for the row that has rear
price <- rear_wheel$Price
mean(price)
```

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

3. What is the minimum horsepower of all cars with capacity for 7 passengers? With a capacity of at least 6 passengers?

```r
cap7 <- Cars93$Passengers == 7
power7 <- Cars93[cap7, ]$Horsepower
min(power7)
```

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

```r
cap6 <- Cars93$Passengers == 6
power6 <- Cars93[cap6, ]$Horsepower
min(power6)
```

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

4. Assuming that these cars are exactly as fuel efficient as this table indicates, find the cars that have the maximum, minimum and median distance travellable for highway driving. You will need at least two columns to work this out; why those two?

```r
travel <- Cars93$MPG.highway * Cars93$Fuel.tank.capacity
max(travel)
```

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

```r
min(travel)
```

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

```r
median(travel)
```

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

To work in this problem we want to know the distance travellable, so we need to get information from 2 columns; MPG.highway and Fuel.tank .capacity. Then we can compute Distance = MPG.highway * Fuel.tank .capacity.

Part II – Reproducibility and Functions

Some of the lectures have included examples of planning production for a factory that turns steel and labor into cars and trucks. Below is a piece of code that optimizes the factory's output (roughly) given the available resources, using a `repeat` loop. It's embedded in a function to make it easier for you to run.

```r
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)
}
```

5. Run the function above with the command to obtain a default output value, starting from a very low initial planned output. What is the final output capacity obtained?

```r
factory.function()
```

```
##     cars   trucks
## 10.29803 19.75496
```

6. Repeat this four more times to obtain new output values. Do these answers differ from each other? If so why? If not, why not?

```r
factory.function(1,1)
```

```
##     cars   trucks
## 10.25846 19.81691
```

```r
factory.function(1,100)
```

```
##      cars    trucks
##  9.738441 20.063746
```

```r
factory.function(100,1)
```

```
##     cars   trucks
## 10.29137 19.78494
```

```r
factory.function(100,100)
```

```
##     cars   trucks
## 10.05189 19.95015
```

```r
factory.function(1000,1000)
```

```
##     cars   trucks
## 10.23190 19.81288
```

These answers are all about the same because the code try to optimize the factory's output. Therefore no matter what vaules we give in, they should converge to the same result; about 10 cars and 20 trucks.

7. Right now, the number of `passes` is a value held within the function itself and not shared. Change the code so that the number of `passes` will be returned at the end of the function, as well as the final `output`.

```r
factory.function1 <- 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))
  }
  giveout <- list("iteration" = passes, "output plan" = output, "demand resourses" = needed)
  return(giveout)
}
```

```r
factory.function1()[1]             # Only want to know number of iteration (passes)
```

```
## $iteration
## [1] 920
```

8. Now, set the initial output levels to 30 cars and 20 trucks and run the code. What is the final output plan (`output`)? What is the final demand for resources (`needed`)? Is the plan within budget and within the slack? How many iterations did it take to converge (`passes`)? For all but `output` you will need to either print this message out deliberately, or return an object that contains all the quantities you want.

```r
factory.function1(30,20)
```

```
## $iteration
## [1] 956
##
## $`output plan`
##     cars   trucks
## 10.32591 19.65911
##
## $`demand resourses`
##             [,1]
## labor 1592.58285
## steel   69.30324
```

As a result, we can see that provided resourses can only produce 10 cars and 20 trucks. Our plan is to produce 30 cars and 20 trucks, that will make it goes over budget.