Applied Statistical Programming - Classes and basic data structures

Messi Lee, Alma Velazquez, Jordan Duffin Wong

1/26/2022

Write the R code to answer the following questions. Write the code and then show what the computer returns when that code is run. Make sure to change the appropriate header in the R code block to make this document compile.

Please put your name at the top of this sheet of paper. You have until the beginning of class 1/31 at 10:00am to answer all of the questions below. You may use R, but not any online documentation. Submit the Rmarkdown and the knitted PDF to Canvas.

1. Change the sign of every odd number in x
   `x <- sample(-100:100, size = 100)`

```
# Code
set.seed(666)
x <- sample(-100:100, size = 100)
print(x)
```

```
##   [1]  -39   25   -5   38   22  -73   41   32    4   39   60  -51   30    0  -88
##  [16]   74   -9  -34   81   19  -43   11  -31   68   27  -33  -29   37   35   61
##  [31]  -74   47  -21  -81   51  -55  -69  -37  -32  -98   33  -71   -6   26   64
##  [46]  -67   89  -53  -89   46  -91   34  -68  -27   72   56  -79 -100    8   98
##  [61]    2   40  -75   -2  -17   90    3  -90   92   21   10   95  -24   71  -18
##  [76]  -64  -54   36   88   76  -84  -48  -57   73  -47   -3  -26   70  -19  -13
##  [91]  -96   78   13   65  -46  -60  -63  -45   96    7
```

```
for (i in 1:length(x)) {
    a <- x[i]%%2
    if (a != 0) {
        x[i] <- x[i] * -1
    }
    # if ((x[i] %% 2) != 0){ x[i] <- x[i] * -1 } print(i)
}
print(x)
```

```
##   [1]   39  -25    5   38   22   73  -41   32    4  -39   60   51   30    0  -88
##  [16]   74    9  -34  -81  -19   43  -11   31   68  -27   33   29  -37  -35  -61
##  [31]  -74  -47   21   81  -51   55   69   37  -32  -98  -33   71   -6   26   64
##  [46]   67  -89   53   89   46   91   34  -68   27   72   56   79 -100    8   98
##  [61]    2   40   75   -2   17   90   -3  -90   92  -21   10  -95  -24  -71  -18
##  [76]  -64  -54   36   88   76  -84  -48   57  -73   47    3  -26   70   19   13
##  [91]  -96   78  -13  -65  -46  -60   63   45   96   -7
```

2. Take the dot product of x and y
   `x <- 1:100`
   `y <- 100:1`

```r
# Code
x <- 1:100
y <- 100:1

z <- x %*% y
```

3. Use the `seq()` and `paste()` to create the vector called `varnames` containing

```
"Var1" "Var2" "Var3" "Var4" "Var5" "Var6"
```

```r
# Code
varnames <- paste("Var", seq(1, 6), sep = "")
print(varnames)
```

```
## [1] "Var1" "Var2" "Var3" "Var4" "Var5" "Var6"
```

4. Remove the substring `"Var"` from the `varnames` vector

```r
# Code
varnames <- gsub("Var", replacement = "", varnames)
print(varnames)
```

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

5. Recast the `varnames` vector into a numeric.

```r
# Code
varnames <- as.numeric(varnames)
print(varnames)
```

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

6. Subset the resulting vector `varnames` to be only odd numbers and make a new vector called `varnames2`

```r
# Code
varnames2 <- varnames[c(TRUE, FALSE)]
print(varnames2)
```

```
## [1] 1 3 5
```

7. If I run the command

```
varnames - varnames2
```

what calculation is being performed?

```r
# Code
varnames - varnames2
```

```
## [1]  0 -1 -2  3  2  1
```

```r
# Recycling varnames2 and doing elementwise subtraction
```

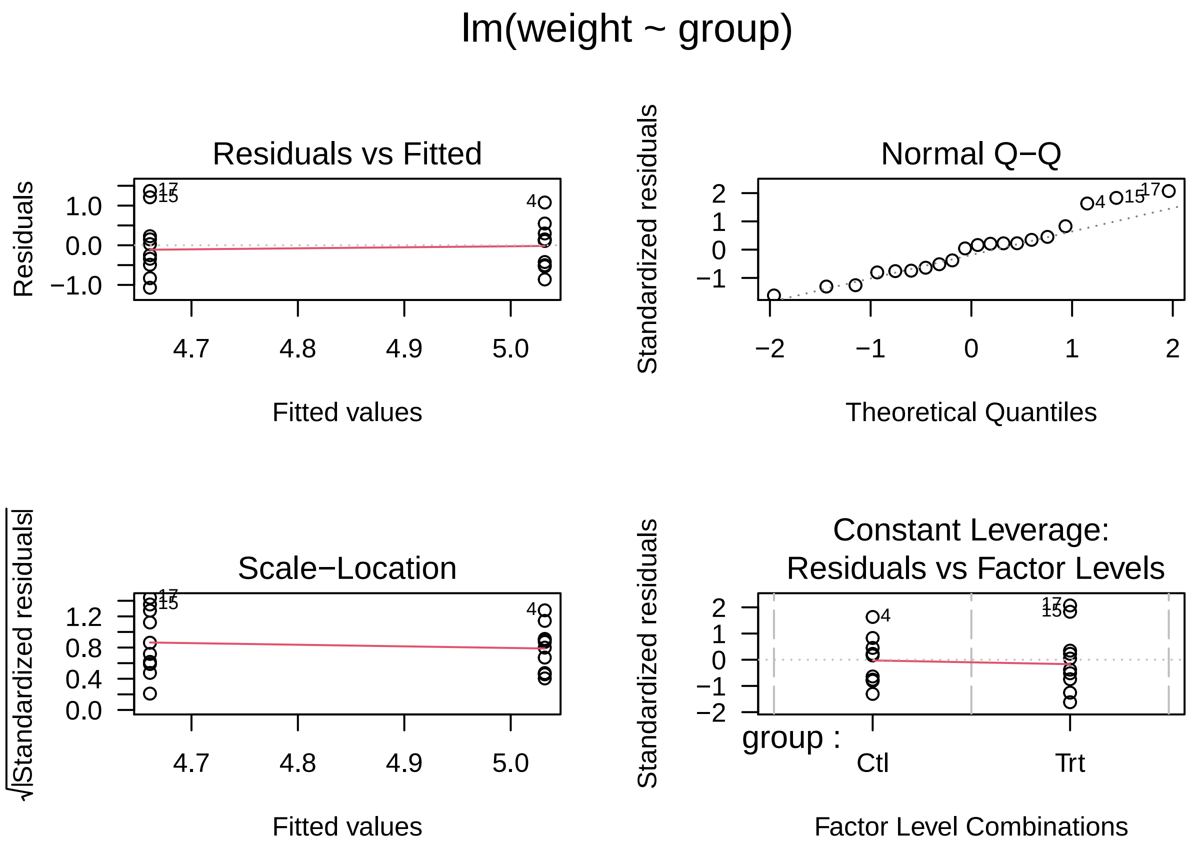
Run the commands below, and then answer the questions listed below.

```r
rm(list=ls())
example(lm)
newLM<-unclass(lm.D90)
```

1. What is the class of element `model` component of `lm.D90`?

```r
# Code
rm(list = ls())  # clearing the environment
example(lm)  # calling the lm example
```

```
##
## lm> require(graphics)
##
## lm> ## Annette Dobson (1990) "An Introduction to Generalized Linear Models".
## lm> ## Page 9: Plant Weight Data.
## lm> ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)
##
## lm> trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)
##
## lm> group <- gl(2, 10, 20, labels = c("Ctl","Trt"))
##
## lm> weight <- c(ctl, trt)
##
## lm> lm.D9 <- lm(weight ~ group)
##
## lm> lm.D90 <- lm(weight ~ group - 1) # omitting intercept
##
## lm> ## No test:
## lm> ##D anova(lm.D9)
## lm> ##D summary(lm.D90)
## lm> ## End(No test)
## lm> opar <- par(mfrow = c(2,2), oma = c(0, 0, 1.1, 0))
##
## lm> plot(lm.D9, las = 1)      # Residuals, Fitted, ...
```



```
##
## lm> par(opar)
##
## lm> ## Don't show:
## lm> ## model frame :
## lm> stopifnot(identical(lm(weight ~ group, method = "model.frame"),
## lm+                     model.frame(lm.D9)))
##
## lm> ## End(Don't show)
## lm> ### less simple examples in "See Also" above
## lm>
## lm>
## lm>
```

```
newLM <- unclass(lm.D90)  # unclassing the lm.D90 thing from the example

class(lm.D90$model)
```

```
## [1] "data.frame"
```

```
# lm.D90$model is a data.frame object
```

2. What is the class of `newLM`? How does this compare with the class of `lm.D90`?

```
# Code
class(lm.D90)
```

```
## [1] "lm"
```

```
class(newLM)
```

```
## [1] "list"
```

```
# the original is an lm object, but using unclass() turned lm.D90 into a list,
# called newLM
```

3. Change the `names` attribute of the `model` component in `lm.D90` to `Var1 Var2`

```
# Code

attributes(lm.D90$model)$names <- c("Var1", "Var2")
# lm.D90$model
```

4. Using matrix algebra commands, calculate the usual OLS estimate (without a constant) for this regression. Compare your results with the coefficients estimated by `lm()`.

Formula: $(X^TX)^{-1}X^Ty$

```
# Code

# original formula: weight ~ group - 1 the -1 REMOVES the intercept, so results
# should match Var1 is weight, Var2 is group

lm.D90$coefficients
```

```
## groupCtl groupTrt
##    5.032    4.661
```

```
dat <- lm.D90$model

ctl <- as.numeric(dat$Var2 == "Ctl")
trt <- as.numeric(dat$Var2 == "Trt")

x_1_1 <- matrix(c(ctl, trt), ncol = 2)
y <- dat$Var1

beta_hat_1_1 <- solve(t(x_1_1) %*% x_1_1) %*% (t(x_1_1) %*% y)
beta_hat_1_1
```

```
##       [,1]
## [1,] 5.032
## [2,] 4.661
```

5. Re-do this calculation, but now include a constant term.

```
# With an intercept, this is a dependent system; 'Treatment' is a free variable
# reg2 <- lm(data = dat, Var1 ~ Var2) reg2$coefficients

# Beause this is a dependent system, only include last 2 variables (treatment
# and intercept) in order for calculation to work; one of the variables is
# redundant
x_1_2 <- matrix(c(x_1_1, rep(1, length(x_1_1[, 1]))), ncol = 3)

beta_hat_1_2 <- solve(t(x_1_2[, 2:3]) %*% x_1_2[, 2:3]) %*% (t(x_1_2[, 2:3]) %*%
    y)
beta_hat_1_2
```

```
##        [,1]
## [1,] -0.371
## [2,]  5.032
```

6. Randomly switch the values of the first ten observations in the `group` objects and re-estimate the linear models.

```
# Code

# Recall that group is Var2
x_2_1 <- x_1_1
x_2_1[c(5, 3, 7, 9), 1] <- x_2_1[c(5, 3, 7, 9), 2]
x_2_1[c(1, 2, 3, 4), 2] <- x_2_1[c(5, 6, 7, 8), 1]

beta_hat_2_1 <- solve(t(x_2_1) %*% x_2_1) %*% (t(x_2_1) %*% y)
beta_hat_2_1
```

```
##          [,1]
## [1,] 3.604118
## [2,] 4.257647
```

```
x_2_2 <- matrix(c(x_2_1, rep(1, length(x_2_1[, 1]))), ncol = 3)

beta_hat_2_2 <- solve(t(x_2_2) %*% x_2_2) %*% (t(x_2_2) %*% y)
beta_hat_2_2
```

```
##           [,1]
## [1,] 0.3022727
## [2,] 0.1303409
## [3,] 4.6776136
```

7. Create a 4 by 2 by 20 array containing the `model` object from all four linear model objects above. Include the appropriate names for each dimension.

```
m1x <- as.factor(ifelse(x_1_1[, 1] == 1, "Ctl", "Trt"))
model1 <- cbind(y, m1x)

m2x <- as.factor(ifelse(x_1_2[, 1] == 1, "Ctl", "Trt"))
model2 <- cbind(y, m2x)

m3x <- as.factor(ifelse(x_2_1[, 1] == 1, "Ctl", "Trt"))
model3 <- cbind(y, m3x)

m4x <- as.factor(ifelse(x_2_2[, 1] == 1, "Ctl", "Trt"))
model4 <- cbind(y, m4x)

model_array <- array(c(model1, model2, model3, model4), c(20, 2, 4), dimnames = list(1:20,
    c("weight", "group"), c("Model 1", "Model 2", "Model 3", "Model 4")))

model_array
```

```
## , , Model 1
##
##    weight group
## 1    4.17     1
## 2    5.58     1
## 3    5.18     1
## 4    6.11     1
## 5    4.50     1
## 6    4.61     1
## 7    5.17     1
## 8    4.53     1
## 9    5.33     1
## 10   5.14     1
## 11   4.81     2
## 12   4.17     2
## 13   4.41     2
## 14   3.59     2
## 15   5.87     2
## 16   3.83     2
## 17   6.03     2
## 18   4.89     2
## 19   4.32     2
## 20   4.69     2
##
## , , Model 2
##
##    weight group
## 1    4.17     1
## 2    5.58     1
## 3    5.18     1
## 4    6.11     1
## 5    4.50     1
## 6    4.61     1
## 7    5.17     1
## 8    4.53     1
## 9    5.33     1
## 10   5.14     1
## 11   4.81     2
## 12   4.17     2
## 13   4.41     2
## 14   3.59     2
## 15   5.87     2
## 16   3.83     2
## 17   6.03     2
## 18   4.89     2
## 19   4.32     2
## 20   4.69     2
##
## , , Model 3
##
##    weight group
## 1    4.17     1
## 2    5.58     1
## 3    5.18     2
## 4    6.11     1
## 5    4.50     2
## 6    4.61     1
## 7    5.17     2
## 8    4.53     1
## 9    5.33     2
## 10   5.14     1
## 11   4.81     2
## 12   4.17     2
## 13   4.41     2
## 14   3.59     2
## 15   5.87     2
## 16   3.83     2
## 17   6.03     2
## 18   4.89     2
## 19   4.32     2
## 20   4.69     2
##
## , , Model 4
##
##    weight group
## 1    4.17     1
## 2    5.58     1
## 3    5.18     2
## 4    6.11     1
## 5    4.50     2
## 6    4.61     1
## 7    5.17     2
## 8    4.53     1
## 9    5.33     2
## 10   5.14     1
## 11   4.81     2
## 12   4.17     2
## 13   4.41     2
## 14   3.59     2
## 15   5.87     2
## 16   3.83     2
## 17   6.03     2
## 18   4.89     2
## 19   4.32     2
## 20   4.69     2
```

8. Create a new list containing the regression coefficients from all four models.

```
coeff_list <- list(beta_hat_1_1, beta_hat_1_2, beta_hat_2_1, beta_hat_2_2)
coeff_list
```

```
## [[1]]
##       [,1]
## [1,] 5.032
## [2,] 4.661
##
## [[2]]
##        [,1]
## [1,] -0.371
## [2,]  5.032
##
## [[3]]
##          [,1]
## [1,] 3.604118
## [2,] 4.257647
```

```
##
## [[4]]
##           [,1]
## [1,] 0.3022727
## [2,] 0.1303409
## [3,] 4.6776136
```