Homework 6

Angela Huynh

12/01/2016

Grade: 44/50 Amazing histograms!!

Question 1

Consider the following very simple genetic model (very simple – don't worry if you're not a geneticist!). A population consists of equal numbers of two sexes: male and female. At each generation men and women are paired at random, and each pair produces exactly two offspring, one male and one female. We are interested in the distribution of height from one generation to the next. Suppose that the height of both children is just the average of the height of their parents, how will the distribution of height change across generations?

Represent the heights of the current generation as a dataframe with two variables, m and f, for the two sexes. We can use rnorm to randomly generate the population at generation 1:

```
# randomly generate population heights at generation 1
pop1 <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
```

The following function takes the data frame pop and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights of the next generation.

```
# returns heights of the next generation
next_gen <- function(pop) {
    pop$m <- sample(pop$m)
    pop$m <- rowMeans(pop)
    pop$f <- pop$m
    pop
}
```

Use the function next_gen to generate nine generations (you already have the first), then use the function hist to plot the distribution of male heights in each generation (this will require multiple calls to hist). The phenomenon you see is called regression to the mean. Provide (at least) minimal decorations such as title and x-axis labels.

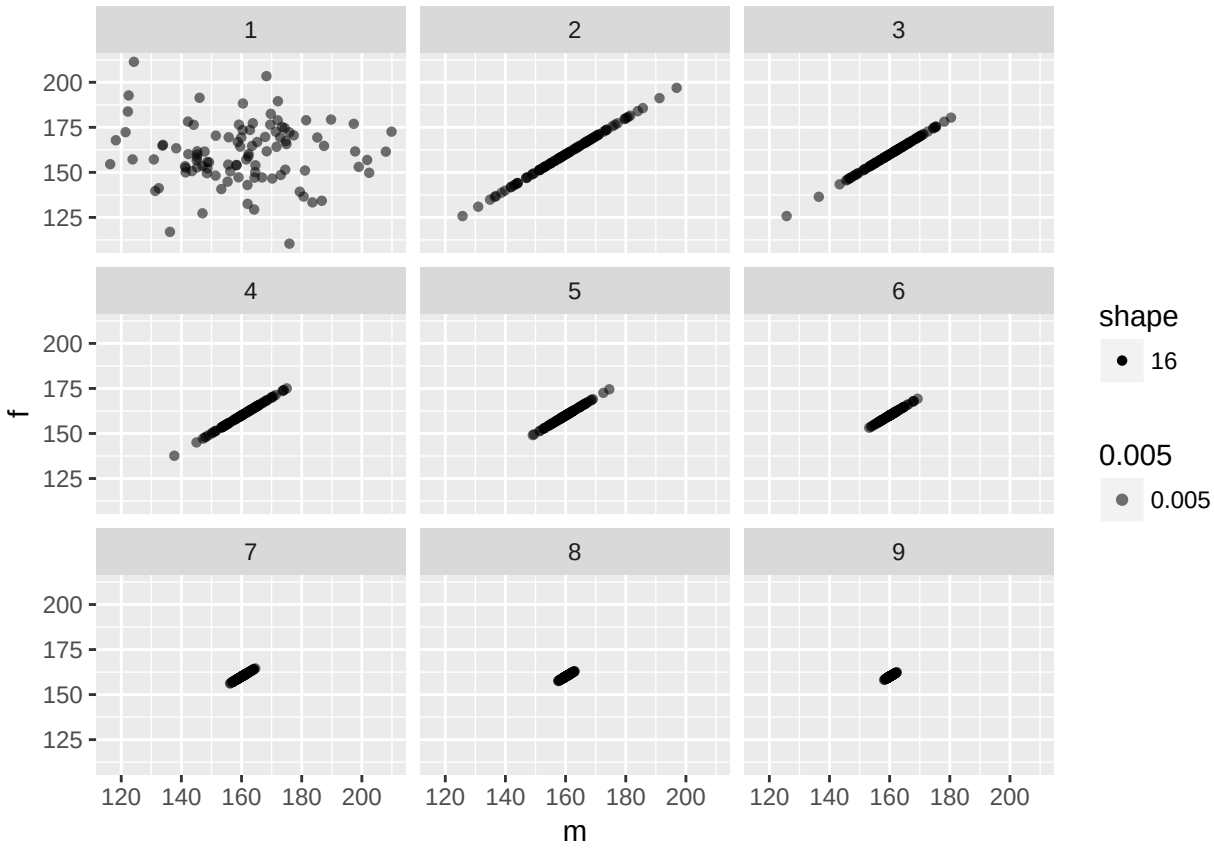
```
# create generations 1-9 1: pop
pop2 <- next_gen(pop1)
pop3 <- next_gen(pop2)
pop4 <- next_gen(pop3)
pop5 <- next_gen(pop4)
pop6 <- next_gen(pop5)
pop7 <- next_gen(pop6)
pop8 <- next_gen(pop7)
pop9 <- next_gen(pop8)

# plot distribution of male heights in each generation
par(mfrow = c(3, 3))
hist(x = pop1$m, xlab = "Height", main = "Histogram of Gen 1 Males", col = "aquamarine")
hist(x = pop2$m, xlab = "Height", main = "Histogram of Gen 2 Males", col = "coral 1")
```

```
hist(x = pop3$m, xlab = "Height", main = "Histogram of Gen 3 Males", col = "cornflower blue")
hist(x = pop4$m, xlab = "Height", main = "Histogram of Gen 4 Males", col = "lavender")
hist(x = pop5$m, xlab = "Height", main = "Histogram of Gen 5 Males", col = "maroon")
hist(x = pop6$m, xlab = "Height", main = "Histogram of Gen 6 Males", col = "dark olive green")
hist(x = pop7$m, xlab = "Height", main = "Histogram of Gen 7 Males", col = "light sky blue")
hist(x = pop8$m, xlab = "Height", main = "Histogram of Gen 8 Males", col = "orange red 2")
hist(x = pop9$m, xlab = "Height", main = "Histogram of Gen 9 Males", col = "violet")
```

Question 2

Use the simulated results from question 1 to reproduce (as closely as possible) the following plot in ggplot2.

```
idx <- sort(rep(1:9, nrow(pop1)))  # create index for respective generation group
gen.data <- cbind(rbind(pop1, pop2, pop3, pop4, pop5, pop6, pop7, pop8, pop9),
    idx)
ggplot(data = gen.data) + geom_point(mapping = aes(x = m, y = f, alpha = 0.005,
    shape = "16")) + facet_wrap(~idx)
```



Question 3

You calculated the power of a study design in question #2 of assignment 3. The study has two variables, treatment group and outcome. There are two treatment groups (0, 1) and they should be assigned randomly with equal probability. The outcome should be a random normal variable with a mean of 60 and standard deviation of 20. If a patient is in the treatment group, add 5 to the outcome.

Starting with a sample size of 250, create a 95% bootstrap percentile interval for the mean of each group. Then create a new bootstrap interval by increasing the sample size by 250 until the sample is 2500. Thus you will create a total of 10 bootstrap intervals. Each bootstrap should create 1000 bootstrap samples. (4 points)

```
set.seed(100)  # set seed to replicate outcome
n <- 1000  # number of times we repeat procedure
alpha <- 0.05

# function to get treatment and outcome data
get.data <- function(sample.size) {
    for (i in (1:sample.size)) {
        treatment <- rbinom(n = sample.size, size = 1, prob = 0.5)
        outcome <- rnorm(n = sample.size, mean = 60, sd = 20)
        for (a in (1:sample.size)) {
            if (treatment[a] == 1) {
                outcome[a] <- outcome[a] + 5
            }
        }
    }
```

```
    study.data <- data.frame(cbind(treatment, outcome))
    return(study.data)
}

# vector of means for two treatment groups

# treatment = 0
means.group0 <- vector()
lower.bounds0 <- vector()
upper.bounds0 <- vector()

# treatment = 1
means.group1 <- vector()
lower.bounds1 <- vector()
upper.bounds1 <- vector()

# bootstrapping
for (i in seq(250, 2500, 250)) {
    current.data <- get.data(i)
    means.0 <- vector()
    means.1 <- vector()
    for (j in 1:n) {
        sample.data <- current.data[sample(nrow(current.data), size = i, replace = TRUE),
            ]
        means.0[j] <- mean(sample.data[which(sample.data$treatment == 0), ]$outcome)
        means.1[j] <- mean(sample.data[which(sample.data$treatment == 1), ]$outcome)
    }
    means.group0 <- c(means.group0, mean(means.0))
    means.group1 <- c(means.group1, mean(means.1))
    lower0 <- quantile(means.0, c(0.025, 0.975))[1]
    upper0 <- quantile(means.0, c(0.025, 0.975))[2]
    lower1 <- quantile(means.1, c(0.025, 0.975))[1]
    upper1 <- quantile(means.1, c(0.025, 0.975))[2]
    lower.bounds0 <- c(lower.bounds0, lower0)
    upper.bounds0 <- c(upper.bounds0, upper0)
    lower.bounds1 <- c(lower.bounds1, lower1)
    upper.bounds1 <- c(upper.bounds1, upper1)
}

# displaying means and bounds in dataframe
bootstrap.0 <- data.frame(means = means.group0, lower = lower.bounds0, upper = upper.bounds0)
bootstrap.0$idx <- 0
bootstrap.0
```

```
##       means    lower    upper idx
## 1  61.87608 58.22476 65.44718   0
## 2  60.64316 58.16243 63.04110   0
## 3  60.92804 58.94679 62.99137   0
## 4  61.35724 59.52067 63.16620   0
## 5  59.73227 58.07035 61.33141   0
## 6  59.77053 58.25908 61.21849   0
## 7  61.36375 59.98328 62.76933   0
## 8  59.62155 58.27375 60.92815   0
## 9  60.16095 58.90425 61.26436   0
```

```
## 10 59.74052 58.62547 60.87157   0
```

```
bootstrap.1 <- data.frame(means = means.group1, lower = lower.bounds1, upper = upper.bounds1)
bootstrap.1$idx <- 1
bootstrap.1
```

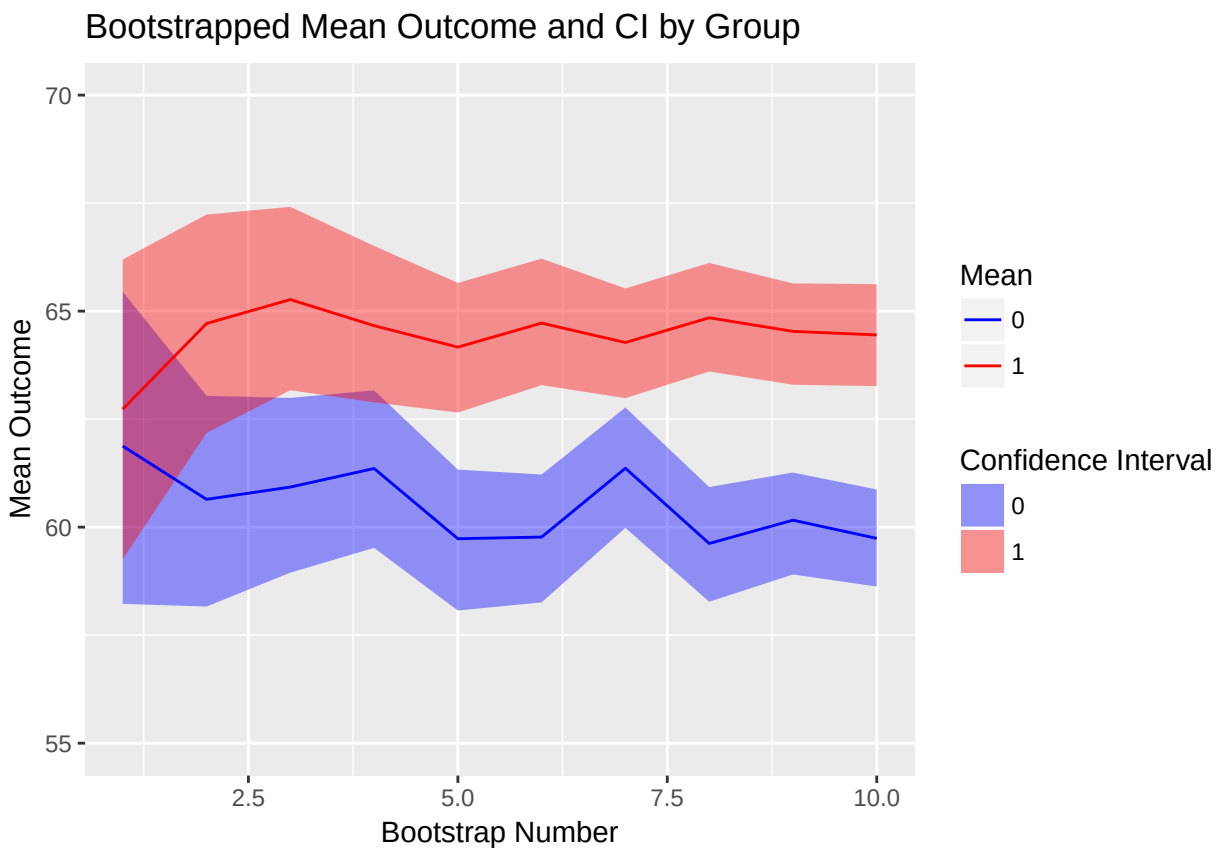
```
##       means    lower    upper idx
## 1  62.73276 59.26031 66.19542   1
## 2  64.71314 62.18047 67.23540   1
## 3  65.26796 63.16932 67.41371   1
## 4  64.66346 62.88724 66.50835   1
## 5  64.16647 62.65259 65.65302   1
## 6  64.72276 63.28618 66.21463   1
## 7  64.27333 62.98603 65.52441   1
## 8  64.84649 63.60313 66.11558   1
## 9  64.53151 63.29663 65.64165   1
## 10 64.44944 63.26356 65.62446   1
```

Produce a line chart that includes the bootstrapped mean and lower and upper percentile intervals for each group. Add appropriate labels and a legend. (6 points)

```
# combine two data frames
bootstrap <- rep(1:10, 2)  # boostrap number 1-10 for each treatment group
all.data <- cbind(bootstrap, rbind(bootstrap.0, bootstrap.1))
all.data
```

```
##    bootstrap    means    lower    upper idx
## 1          1 61.87608 58.22476 65.44718   0
## 2          2 60.64316 58.16243 63.04110   0
## 3          3 60.92804 58.94679 62.99137   0
## 4          4 61.35724 59.52067 63.16620   0
## 5          5 59.73227 58.07035 61.33141   0
## 6          6 59.77053 58.25908 61.21849   0
## 7          7 61.36375 59.98328 62.76933   0
## 8          8 59.62155 58.27375 60.92815   0
## 9          9 60.16095 58.90425 61.26436   0
## 10        10 59.74052 58.62547 60.87157   0
## 11         1 62.73276 59.26031 66.19542   1
## 12         2 64.71314 62.18047 67.23540   1
## 13         3 65.26796 63.16932 67.41371   1
## 14         4 64.66346 62.88724 66.50835   1
## 15         5 64.16647 62.65259 65.65302   1
## 16         6 64.72276 63.28618 66.21463   1
## 17         7 64.27333 62.98603 65.52441   1
## 18         8 64.84649 63.60313 66.11558   1
## 19         9 64.53151 63.29663 65.64165   1
## 20        10 64.44944 63.26356 65.62446   1
```

```
# line chart with bootstrap mean and lower/upper percentile intervals for
# each group
p <- ggplot(data = all.data, aes(x = bootstrap, y = means)) + scale_fill_manual(values = c("blue",
    "red"), name = "Confidence Interval") + scale_colour_manual(values = c("blue",
    "red"), name = "Mean") + geom_ribbon(aes(ymin = all.data$lower, ymax = all.data$upper,
    group = factor(idx), fill = factor(idx)), alpha = 0.4) + geom_line(aes(group = factor(idx),
    colour = factor(idx))) + ylab("Mean Outcome") + xlab("Bootstrap Number") +
    ggtitle("Bootstrapped Mean Outcome and CI by Group") + ylim(55, 70)
p
```



Question 4

Programming with classes. The following function will generate random patient information.

```
makePatient <- function() {
    vowel <- grep("[aeiou]", letters)
    cons <- grep("[^aeiou]", letters)
    name <- paste(sample(LETTERS[cons], 1), sample(letters[vowel], 1), sample(letters[cons],
        1), sep = "")
    gender <- factor(sample(0:1, 1), levels = 0:1, labels = c("female", "male"))
    dob <- as.Date(sample(7500, 1), origin = "1970-01-01")
    n <- sample(6, 1)
    doa <- as.Date(sample(1500, n), origin = "2010-01-01")
    pulse <- round(rnorm(n, 80, 10))
    temp <- round(rnorm(n, 98.4, 0.3), 2)
    fluid <- round(runif(n), 2)
    list(name, gender, dob, doa, pulse, temp, fluid)
}
```

1. Create an S3 class medicalRecord for objects that are a list with the named elements name, gender, date_of_birth, date_of_admission, pulse, temperature, fluid_intake. Note that an individual patient may have multiple measurements for some measurements. Set the RNG seed to 8 and create a medical record by taking the output of makePatient. Print the medical record, and print the class of the medical record. (5 points)

```r
set.seed(8)

# creation of S3 class medicalRecord
new.patient <- makePatient()
names(new.patient) <- c("name", "gender", "dob", "doa", "pulse", "temp", "fluid")

class(new.patient)
```

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

```r
class(new.patient) <- "medicalRecord"  # set class to medicalRecord
new.patient
```

```
## $name
## [1] "Mev"
##
## $gender
## [1] male
## Levels: female male
##
## $dob
## [1] "1976-08-09"
##
## $doa
## [1] "2011-03-14" "2013-10-30" "2013-02-27" "2012-08-23" "2011-11-16"
##
## $pulse
## [1] 67 81 95 74 81
##
## $temp
## [1] 98.33 98.16 99.00 98.49 98.67
##
## $fluid
## [1] 0.62 0.93 0.18 0.39 0.34
##
## attr(,"class")
## [1] "medicalRecord"
```

```r
class(new.patient)
```

```
## [1] "medicalRecord"
```

2. Write a medicalRecord method for the generic function mean, which returns averages for pulse, temperature and fluids. Also write a medicalRecord method for print, which employs some nice formatting, perhaps arranging measurements by date, and plot that generates a composite plot of measurements over time. Call each function for the medical record created in part 1. (5 points)

```r
# medicalRecord method for generic function mean rounded 2 decimal places
mean.medicalRecord <- function(patient) {
    pulses <- as.numeric(unlist(patient["pulse"]))
    temps <- as.numeric(unlist(patient["temp"]))
    fluids <- as.numeric(unlist(patient["fluid"]))
    mean.pulses <- mean(pulses)
    mean.temps <- mean(temps)
    mean.fluids <- mean(fluids)
    cat(sprintf("Average Pulse: %.2f \nAverage temperature: %.2f \nFluid: %.2f",
```

```
        mean.pulses, mean.temps, mean.fluids), "\n")
}

mean.medicalRecord(new.patient)
```

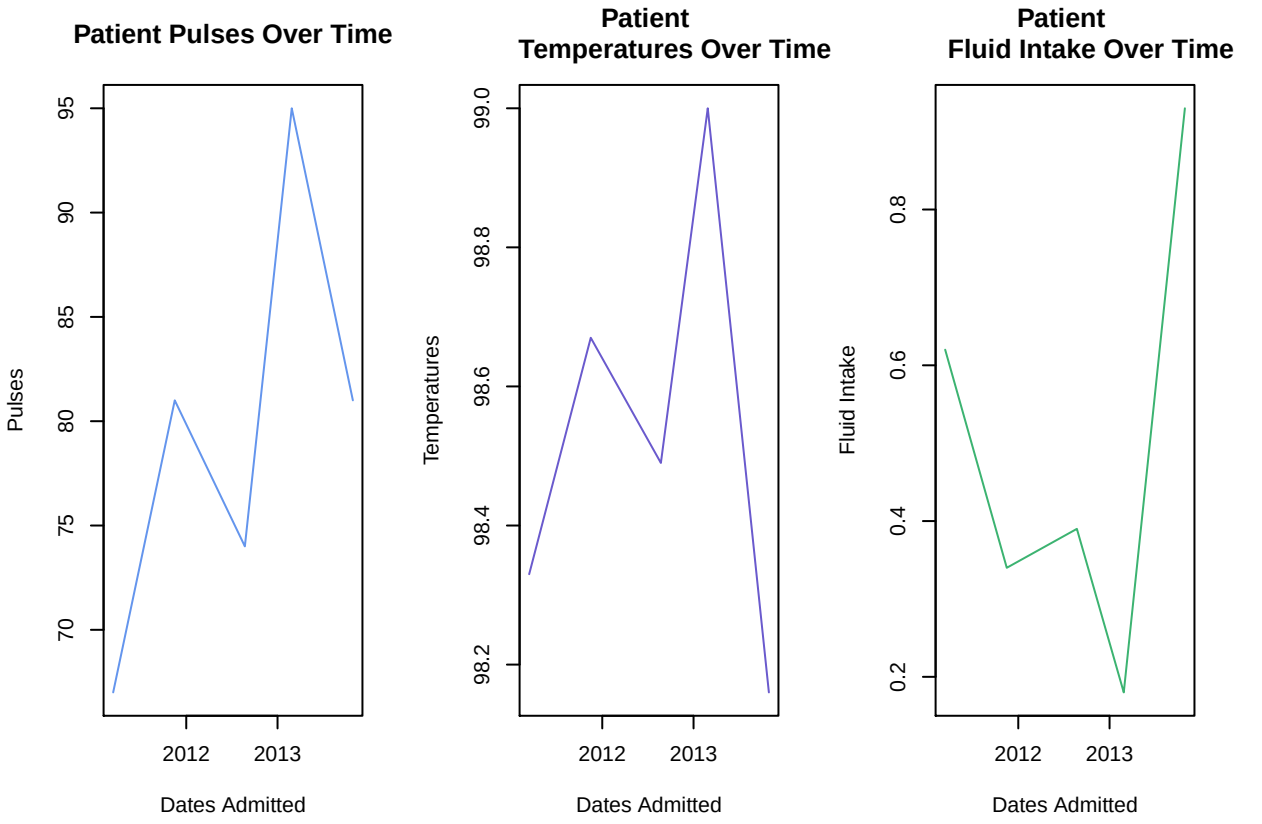
```
## Average Pulse: 79.60
## Average temperature: 98.53
## Fluid: 0.49
```

```
# medicalRecord method for print

print.patient <- function(patient) {
    name <- as.character(unlist(patient["name"]))
    gender <- as.character(unlist(patient["gender"]))
    date.birth <- as.Date(unlist(patient["dob"]), origin = "1970-01-01")
    dates.admitted <- as.Date(unlist(patient["doa"]), origin = "1970-01-01")
    pulses <- as.numeric(unlist(patient["pulse"]))
    temps <- as.numeric(unlist(patient["temp"]))
    fluids <- as.numeric(unlist(patient["fluid"]))
    patient.data <- data.frame(dates.admitted, pulses, temps, fluids)
    ordered.data <- patient.data[order(as.Date(patient.data$dates, "%d/%m/%Y"),
        decreasing = FALSE), ]
    cat(sprintf("Patient name: %s \nGender: %s \nDate of Birth: %s \n", name,
        gender, date.birth))
    print(ordered.data)
}

print.patient(new.patient)
```

```
## Patient name: Mev
## Gender: male
## Date of Birth: 1976-08-09
##      dates.admitted pulses temps fluids
## doa1     2011-03-14     67 98.33   0.62
## doa5     2011-11-16     81 98.67   0.34
## doa4     2012-08-23     74 98.49   0.39
## doa3     2013-02-27     95 99.00   0.18
## doa2     2013-10-30     81 98.16   0.93
```

```
# plotting of measurements
plot.patient <- function(patient) {
    dates.admitted <- as.Date(unlist(patient["doa"]), origin = "1970-01-01")
    pulses <- as.numeric(unlist(patient["pulse"]))
    temps <- as.numeric(unlist(patient["temp"]))
    fluids <- as.numeric(unlist(patient["fluid"]))
    patient.data <- data.frame(dates.admitted, pulses, temps, fluids)
    ordered.data <- patient.data[order(as.Date(patient.data$dates, "%d/%m/%Y"),
        decreasing = FALSE), ]
    par(mfrow = c(1, 3))
    plot(x = ordered.data$dates.admitted, y = ordered.data$pulses, type = "l",
        main = "Patient Pulses Over Time", xlab = "Dates Admitted", ylab = "Pulses",
        col = "cornflower blue")
    plot(x = ordered.data$dates.admitted, y = ordered.data$temps, type = "l",
        main = "Patient
       Temperatures Over Time", xlab = "Dates Admitted",
```

```
        ylab = "Temperatures", col = "slate blue")
    plot(x = ordered.data$dates.admitted, y = ordered.data$fluids, type = "l",
        main = "Patient
       Fluid Intake Over Time", xlab = "Dates Admitted",
        ylab = "Fluid Intake", col = "medium sea green")
}

plot.patient(new.patient)
```



JC Grading -2 On the right track ... the mean method was done correctly (mean.medicalRecord <- function(obj){ ... }). The print and plot methods should be written similarly with the medicalRecord suffix (print.medicalRecord <- function(obj){} and plot.medicalRecord <- function(obj) {})

3. Create a further class for a cohort (group) of patients, and write methods for mean and print which, when applied to a cohort, apply mean or print to each patient contained in the cohort. Hint: think of this as a "container" for patients. Reset the RNG seed to 8 and create a cohort of ten patients, then show the output for mean and print. (5 points)

```
set.seed(8)

# class for cohort of 10 patients
cohort <- replicate(10, makePatient())
rownames(cohort) <- c("name", "gender", "dob", "doa", "pulse", "temp", "fluid")

# mean method for cohort
mean.10 <- function(patients) {
    mean.pulses <- vector()
    mean.temps <- vector()
```

```
    mean.fluids <- vector()
    for (i in 1:10) {
        mean.pulses[i] <- mean(as.numeric(unlist(patients[5, i])))
        mean.temps[i] <- mean(as.numeric(unlist(patients[6, i])))
        mean.fluids[i] <- mean(as.numeric(unlist(patients[7, i])))
    }
    means.all <- data.frame(mean.pulses, mean.temps, mean.fluids)
    print(means.all)
}

mean.10(cohort)  # output of mean
```

```
##    mean.pulses mean.temps mean.fluids
## 1     79.60000   98.53000   0.4920000
## 2     78.00000   98.49500   0.2450000
## 3     81.50000   98.44000   0.4033333
## 4     78.00000   98.60000   0.6500000
## 5     88.33333   98.05000   0.5866667
## 6     83.50000   98.45000   0.4525000
## 7     83.00000   98.01000   0.9700000
## 8     77.50000   98.14833   0.3366667
## 9     77.00000   98.83000   0.4450000
## 10    79.33333   98.30000   0.6583333
```

```
print.10 <- function(patients) {
    for (i in 1:10) {
        name <- as.character(unlist(patients["name", i]))
        gender <- as.character(unlist(patients["gender", i]))
        date.birth <- as.Date(unlist(patients["dob", i]), origin = "1970-01-01")
        dates.admitted <- as.Date(unlist(patients["doa", i]), origin = "1970-01-01")
        pulses <- as.numeric(unlist(patients["pulse", i]))
        temps <- as.numeric(unlist(patients["temp", i]))
        fluids <- as.numeric(unlist(patients["fluid", i]))
        patient.data <- data.frame(dates.admitted, pulses, temps, fluids)
        ordered.data <- patient.data[order(as.Date(patient.data$dates, "%d/%m/%Y"),
            decreasing = FALSE), ]
        cat(sprintf("\n"))
        cat(sprintf("Patient name: %s \nGender: %s \nDate of Birth: %s \n",
            name, gender, date.birth))
        print(ordered.data)
    }
}

print.10(cohort)  # output of print
```

```
##
## Patient name: Mev
## Gender: male
## Date of Birth: 1976-08-09
##      dates.admitted pulses temps fluids
## doa1     2011-03-14     67 98.33   0.62
## doa5     2011-11-16     81 98.67   0.34
## doa4     2012-08-23     74 98.49   0.39
## doa3     2013-02-27     95 99.00   0.18
```

```
## doa2     2013-10-30     81 98.16   0.93
##
## Patient name: Yul
## Gender: male
## Date of Birth: 1988-06-28
##      dates.admitted pulses temps fluids
## doa1     2012-01-16     76 98.92   0.14
## doa2     2013-08-07     80 98.07   0.35
##
## Patient name: Zet
## Gender: female
## Date of Birth: 1970-06-13
##      dates.admitted pulses temps fluids
## doa6     2010-03-21     79 98.58   0.22
## doa5     2010-04-01     73 98.32   0.61
## doa4     2012-08-29     88 98.47   0.59
## doa3     2013-06-01     84 98.22   0.25
## doa1     2013-11-03     72 98.54   0.03
## doa2     2014-02-05     93 98.51   0.72
##
## Patient name: Qih
## Gender: female
## Date of Birth: 1987-08-30
##     dates.admitted pulses temps fluids
## doa     2011-06-22     78  98.6   0.65
##
## Patient name: Wut
## Gender: male
## Date of Birth: 1974-06-28
##      dates.admitted pulses temps fluids
## doa3     2010-04-12     76 98.05   0.65
## doa1     2011-02-16     93 98.26   0.97
## doa2     2012-04-12     96 97.84   0.14
##
## Patient name: Juy
## Gender: male
## Date of Birth: 1983-06-09
##      dates.admitted pulses temps fluids
## doa4     2010-03-10     81 99.11   0.66
## doa1     2010-03-25     90 98.58   0.26
## doa3     2010-04-18     75 98.58   0.60
## doa2     2010-06-10     88 97.53   0.29
##
## Patient name: God
## Gender: female
## Date of Birth: 1990-02-12
##     dates.admitted pulses temps fluids
## doa     2010-03-12     83 98.01   0.97
##
## Patient name: Fut
## Gender: male
## Date of Birth: 1970-01-11
##      dates.admitted pulses temps fluids
## doa5     2011-04-07     80 97.87   0.36
```

```
## doa4     2011-04-14     83 97.91   0.00
## doa2     2011-08-16     66 98.49   0.13
## doa1     2013-03-15     74 98.38   0.31
## doa6     2013-06-20     74 98.41   0.49
## doa3     2013-11-12     88 97.83   0.73
##
## Patient name: Pet
## Gender: male
## Date of Birth: 1979-01-01
##      dates.admitted pulses temps fluids
## doa1     2010-10-30     85 98.84   0.60
## doa2     2012-05-10     69 98.82   0.29
##
## Patient name: Yed
## Gender: male
## Date of Birth: 1977-11-11
##      dates.admitted pulses temps fluids
## doa4     2010-01-28     63 97.95   0.94
## doa3     2010-03-06     81 98.45   0.67
## doa1     2010-07-10     98 98.65   0.79
## doa6     2010-08-27     66 97.68   0.36
## doa5     2011-06-18     83 98.00   0.69
## doa2     2013-01-06     85 99.07   0.50
```

JC Grading -4 After creating cohort as you did above, write cohort <- class("cohort") (i.e. assign it to a class now defined as cohort). Then your functions should be mean.cohort and print.cohort. When you call the mean and print functions on cohort it is then mean(cohort) (as compared to mean.10(cohort)).

Also, I did not take off for hard-coding 10 records in your function, but try in your functions to write them so they can take any number of records. You could do something like sizs <- length(cohort).