Bios 301: Assignment 4

Utsav Kumar

Due Tuesday, 2 December, 1:00 PM

$5^{n=day}$ points taken off for each day late.

50 points total.

Submit a single knitr file (named `homework4.rmd`), along with a valid PDF output file. Inside the file, clearly indicate which parts of your responses go with which problems (you may use the original homework document as a template). Add your name as `author` to the file's metadata section. Raw R code/output or word processor files are not acceptable.

Failure to name file `homework4.rmd` or include author name may result in 5 points taken off.

Question 1

15 points

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:

```
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
```

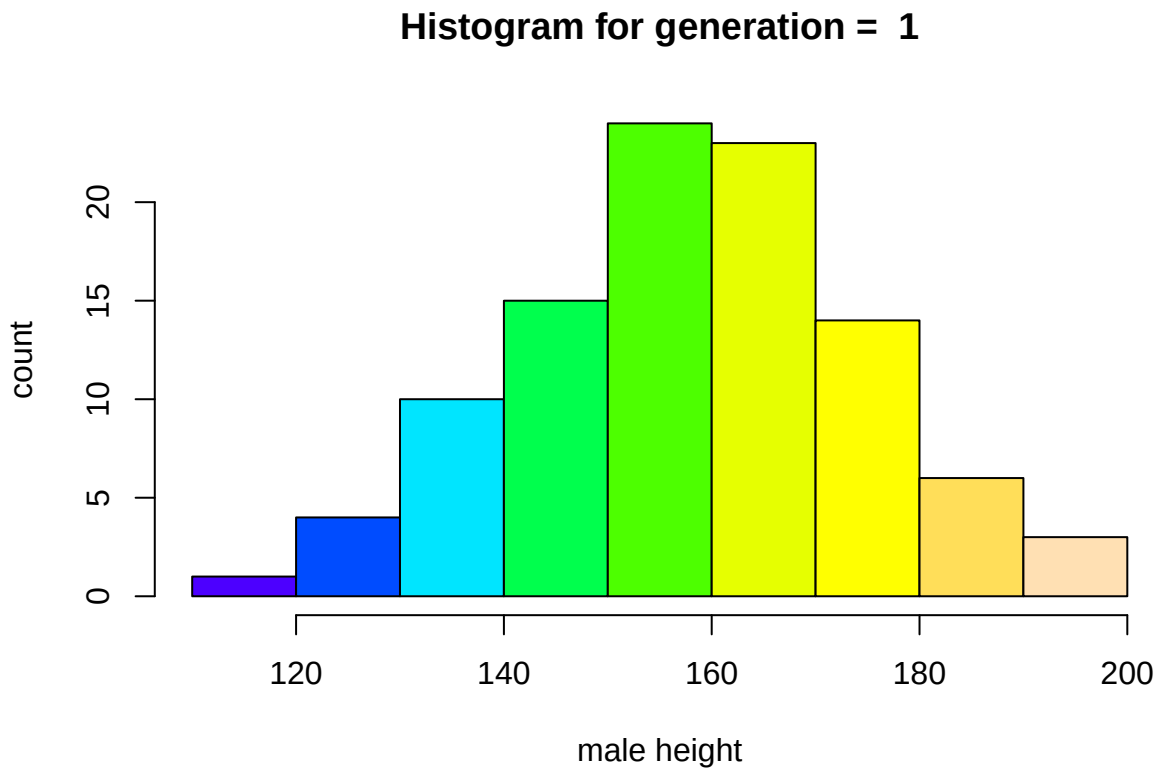
The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights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

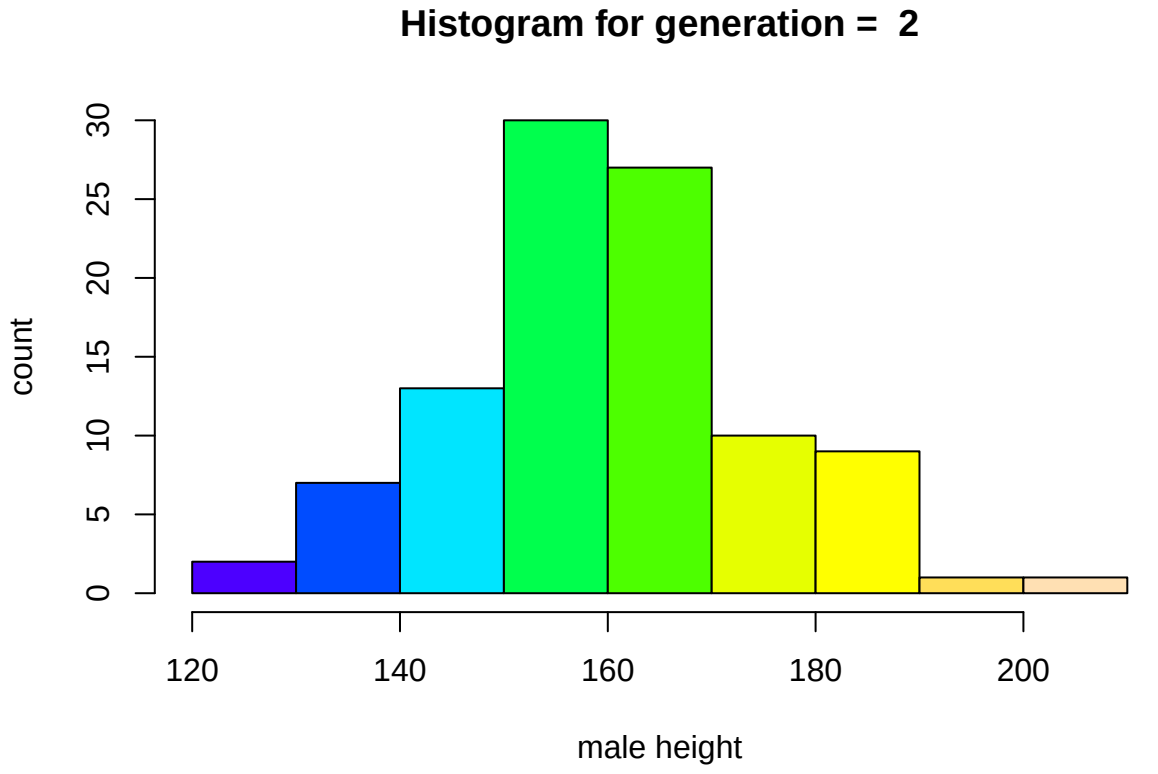
```
next_gen <- function(pop) {
pop$m <- sample(pop$m)
pop$m <- rowMeans(pop)
pop$f <- pop$m
pop
}
```

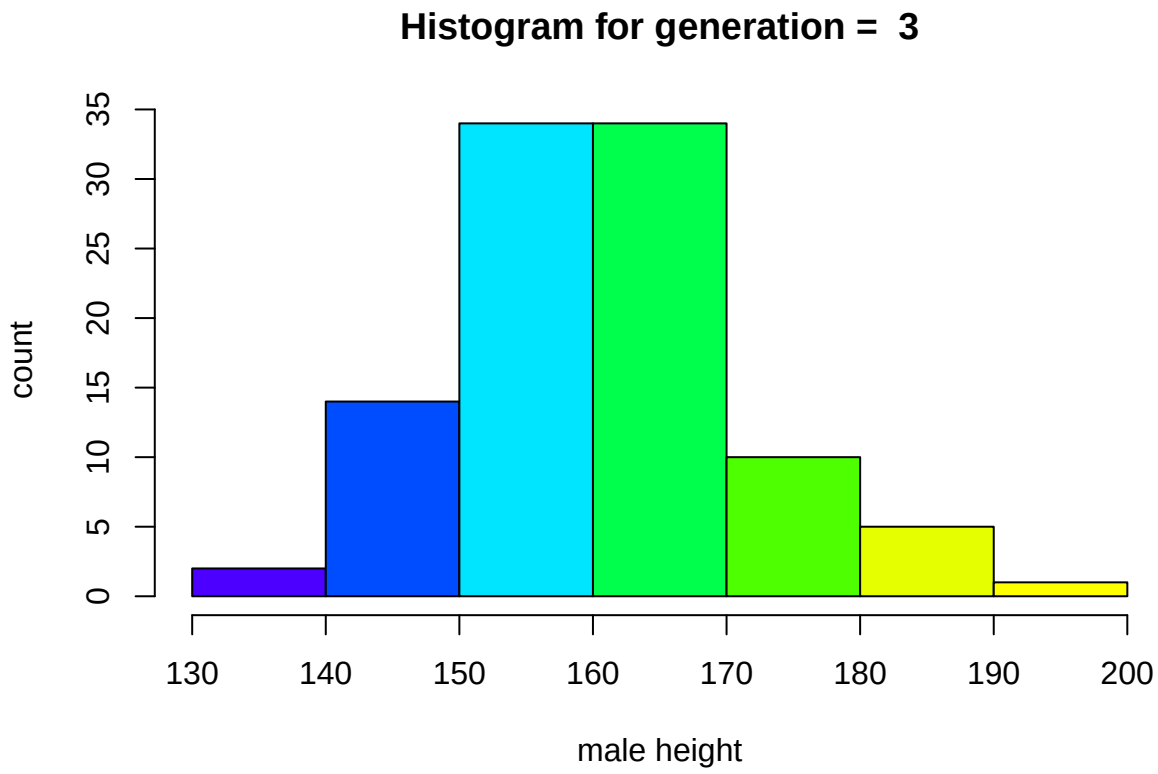
Use the function `next_gen` to generate nine generations (you already have the first), then use the function `hist` to plot the distribution of male heights in each generation (this will require multiple calls to `hist`). The phenomenon you see is called regression to the mean. Provide (at least) minimal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

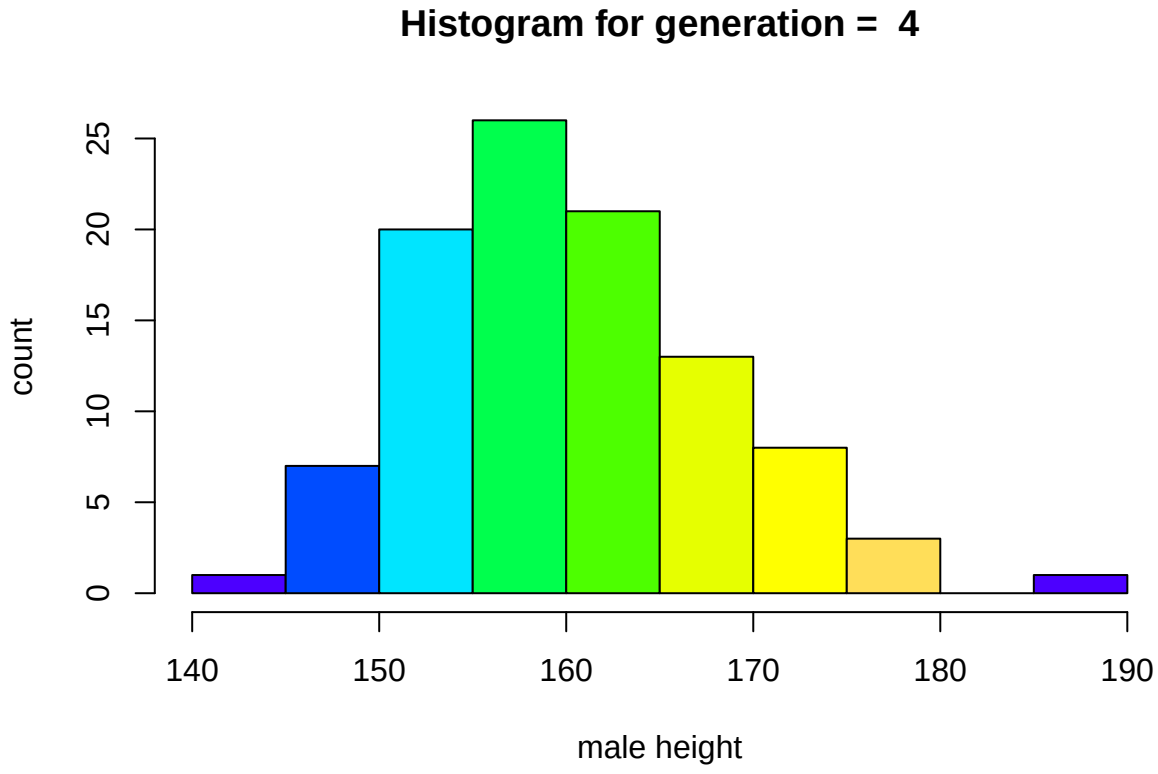
```
# The next gen function

next_gen <- function(pop)
{
  pop$m <-sample(pop$m)
  pop$m <-rowMeans(pop)
```

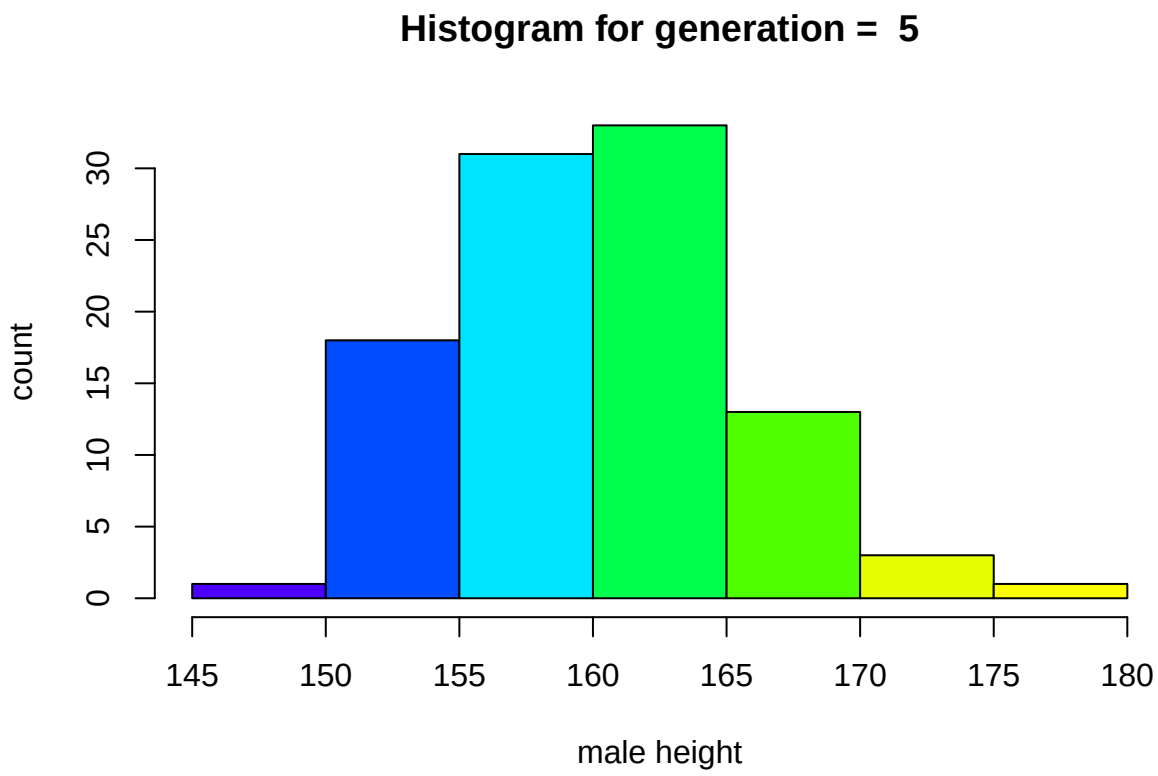
```
  pop$f <- pop$m
  pop
}

# Initializing the 1st generation
pop <- data.frame(m =rnorm(100, 160, 20), f =rnorm(100, 160, 20))
generation <- c(rep(1, nrow(pop)))
generation.pop <- cbind(generation,pop)

# Looping over to find the nex generation using the next_gen function
for (i in seq(2,10))
{
  pop <- next_gen(pop)
  generation <- c(rep(i, nrow(pop)))
  generation.dummy <- cbind(generation,pop)
  generation.pop <- rbind(generation.pop, generation.dummy)
}

# plotting histogram for each generation separately
# Alternately the plots can also be saved using format_type("plot.format_extension")
par(mfrow=c(1,1))
for (i in seq(1:9))
  {
    data <- subset(generation.pop, generation.pop$generation == i)
    data <- data[['m']]
    main = paste("Histogram for generation = ", i)
    hist(data,
         xlab = "male height",
         ylab = "count",
         main = main,
         col = topo.colors(9, alpha = 1))
    cat ("\n\n")
  }
```

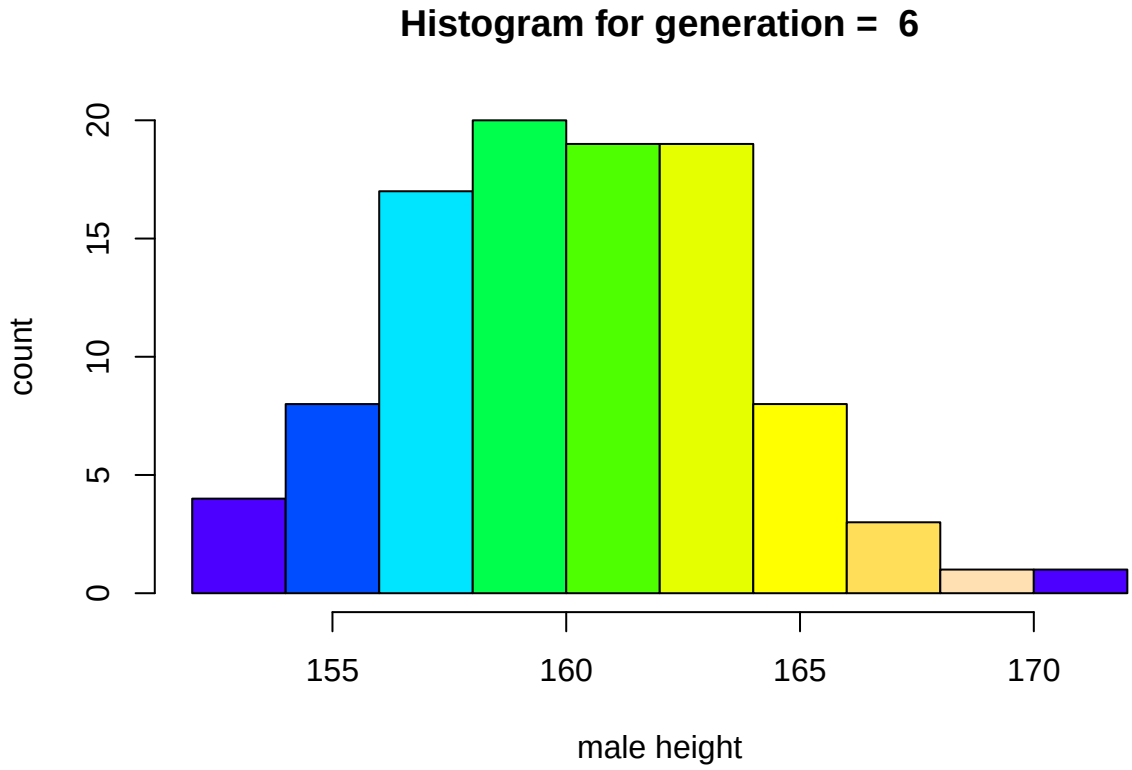

Histogram for generation = 1
count
0
5
10
15
20
120
140
160
180
200
male height

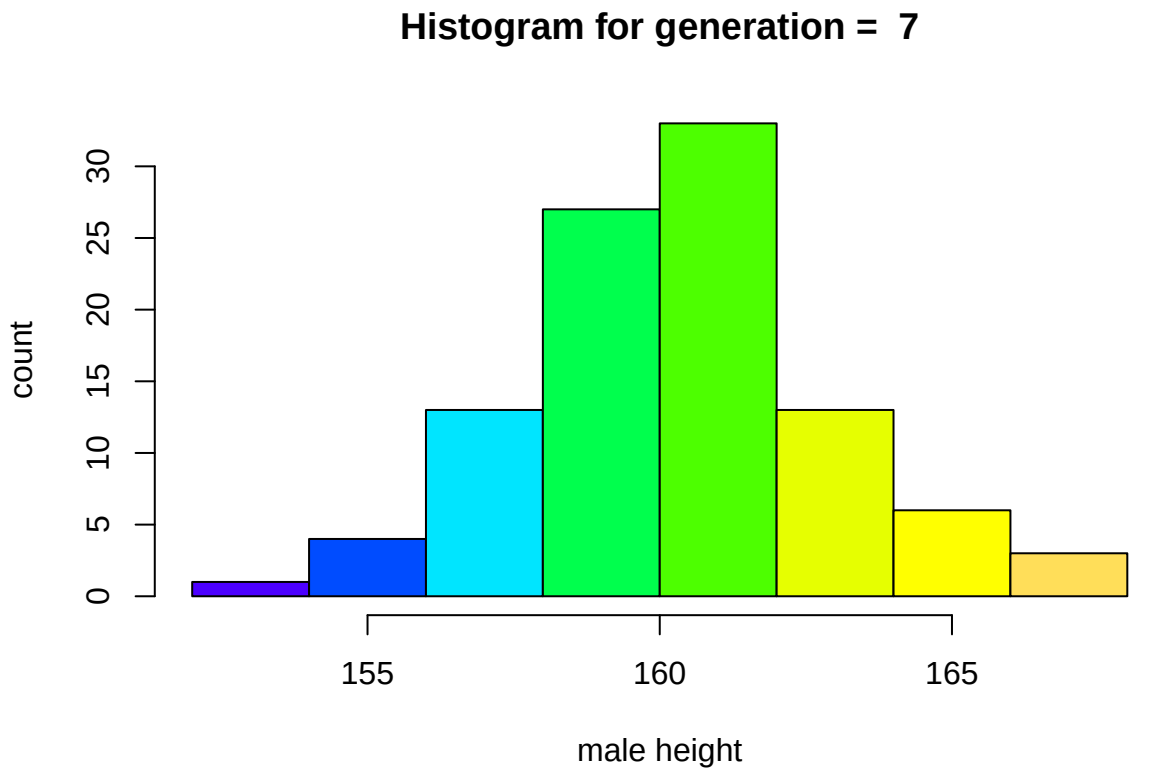

Histogram for generation = 2
count
0
5
10
15
20
25
30
120
140
160
180
200
male height

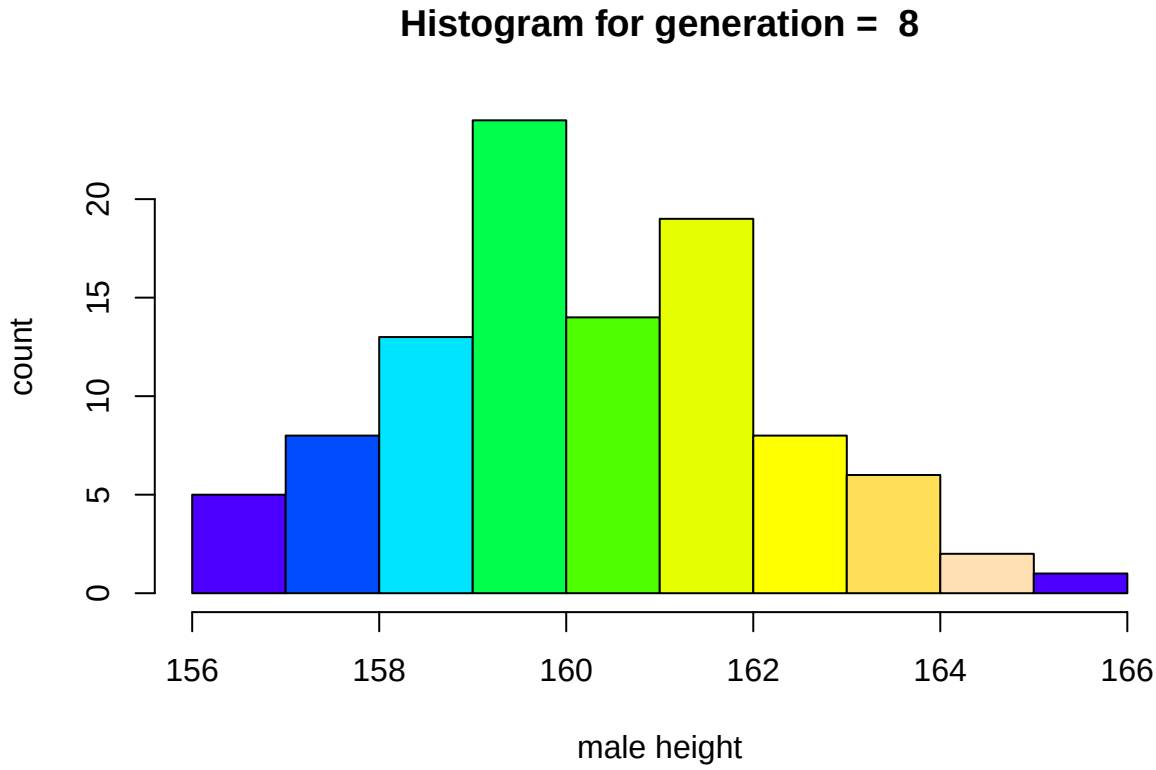

Histogram for generation = 3
count
0
5
10
15
20
25
30
35
130
140
150
160
170
180
190
200
male height

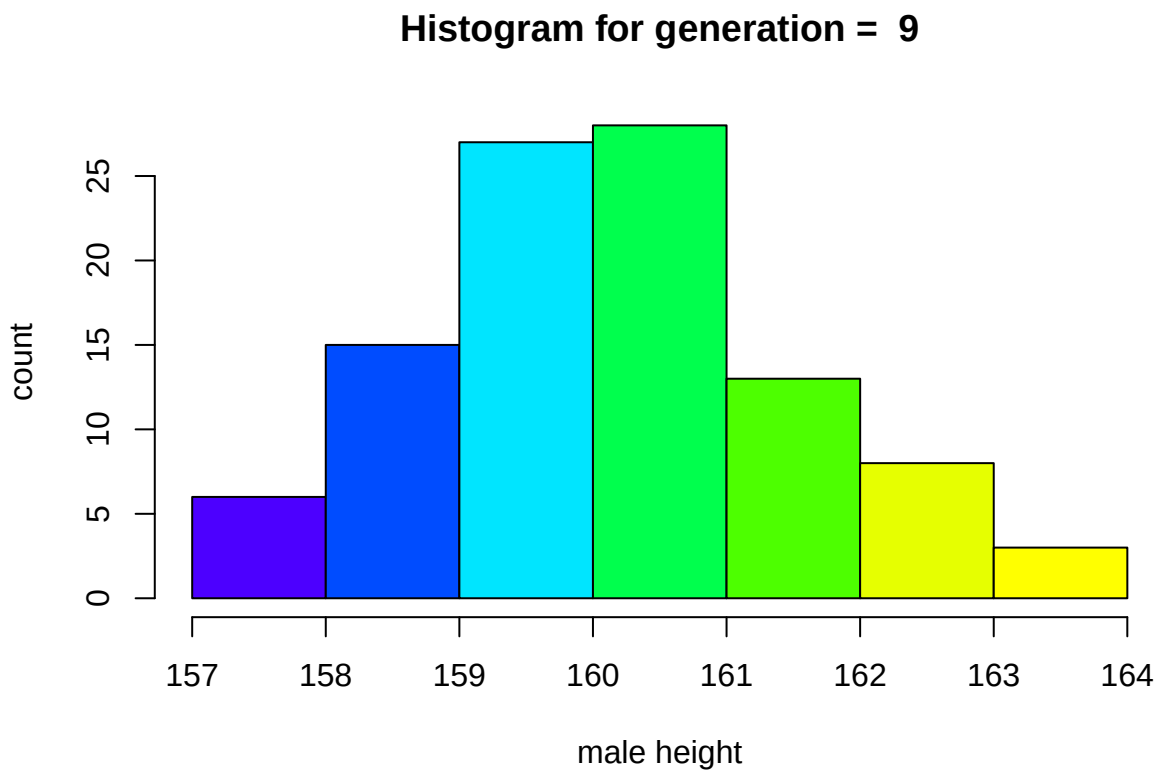

Histogram for generation = 4
count
0
5
10
15
20
25
140
150
160
170
180
190
male height


Histogram for generation = 5
count
0
5
10
15
20
25
30
145
150
155
160
165
170
175
180
male height


Histogram for generation = 6
count
0
5
10
15
20
155
160
165
170
male height


Histogram for generation = 7
count
0
5
10
15
20
25
30
155
160
165
male height


Histogram for generation = 8
count
0
5
10
15
20
156
158
160
162
164
166
male height

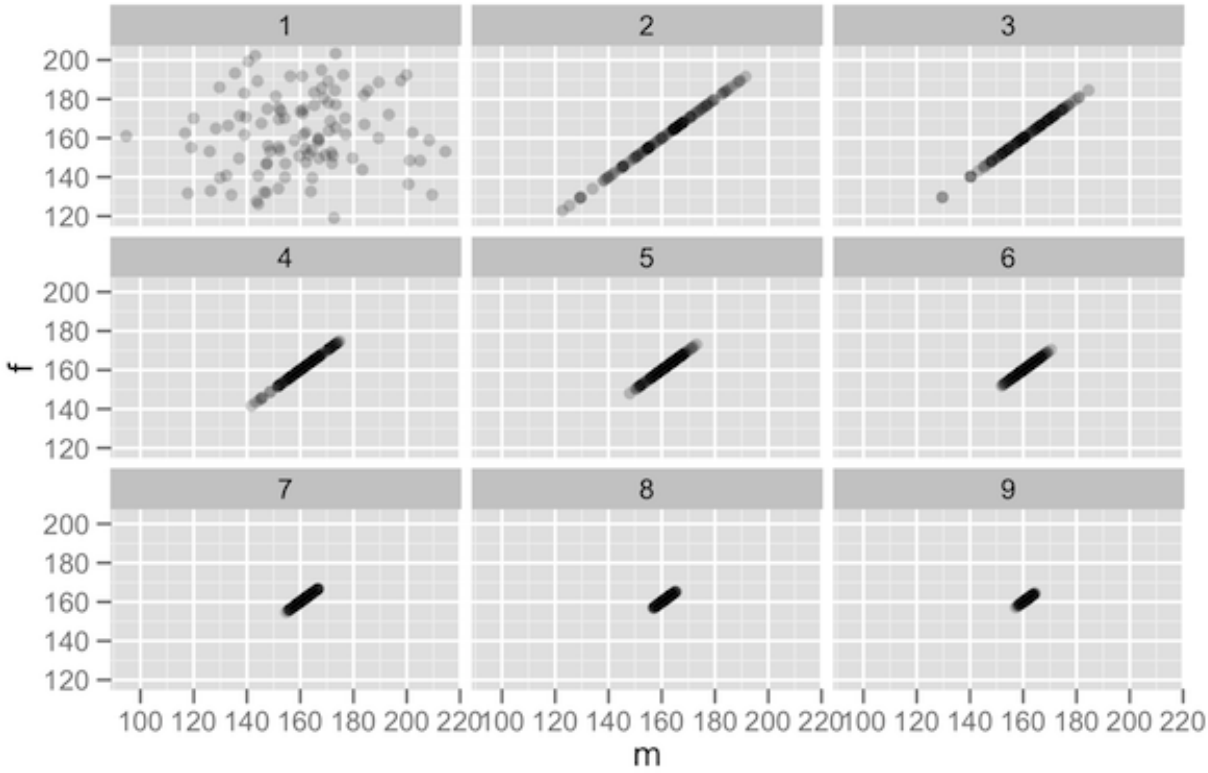



```
# plotting histogram using ggplot
#library (ggplot2)
#p <- ggplot(subset(generation.pop, generation < 10), aes(x=m), main = "Histogram of male height for al
  #geom_histogram(binwidth = 15) + facet_wrap(~generation, nrow = 3, ncol =3)
#p
```

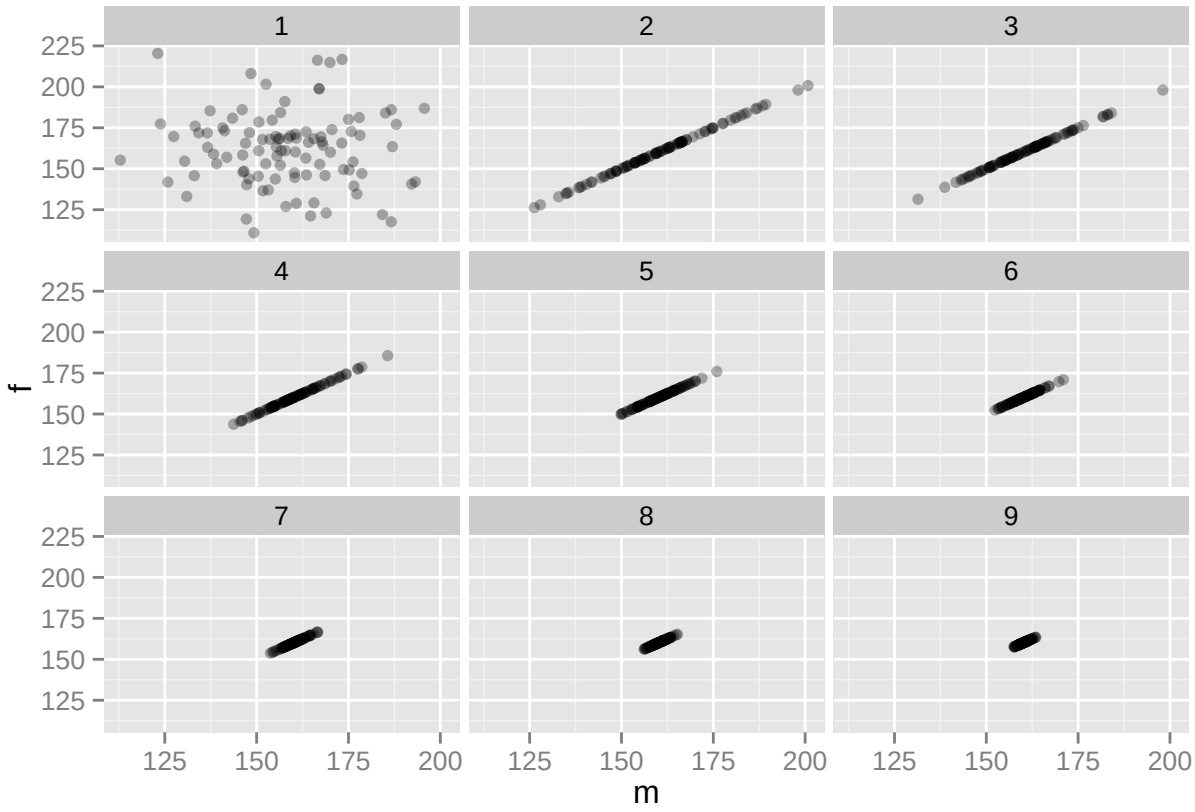
Question 2

10 points

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




```
# plotting simulated results as mentioned in the question

library (ggplot2)
data <- subset(generation.pop, generation < 10)
generation.plot <- ggplot(data, aes(x=m, y=f)) +
  geom_point(alpha=0.3) +
  facet_wrap(~generation, nrow = 3, ncol = 3) +
  labs(x = "m", y = "f")

generation.plot
```



Question 3

10 points

We know that the *U(-1,1)* random variable has mean 0. Use a sample of size 100 to estimate the mean and give a 95% confidence interval. Does the confidence interval contain 0? Repeat the above a large number of times (say, 1000) and set the RNG seed to `1000`. What percentage of time does the confidence interval contain 0? Write your code so that it produces output similar to the following (to save space, only output the first ten trials):

```
Number of trials: 10

Sample mean  lower bound  upper bound  contains mean
-0.0733      -0.1888       0.0422              1
-0.0267      -0.1335       0.0801              1
-0.0063      -0.1143       0.1017              1
-0.0820      -0.1869       0.0230              1
-0.0354      -0.1478       0.0771              1
-0.0751      -0.1863       0.0362              1
-0.0742      -0.1923       0.0440              1
 0.0071      -0.1011       0.1153              1
 0.0772      -0.0322       0.1867              1
-0.0243      -0.1370       0.0885              1

100 percent of CI's contained the mean
```

Hint: the standard deviation for a uniform distribution is (b-a)/sqrt(12).

```r
# for 100 sample cases tested for 1 trial run for 95% confidence interval

set.seed(1000)
i = 1
sample <- runif (100, -1, 1)
l.sample <- length(sample)
mean.sample <- mean(sample)
sd.sample <- sd(sample)

error <- qt(0.975, df=l.sample-1)*sd.sample/sqrt(l.sample)
left <- mean.sample - error
right <- mean.sample + error

data.test <- data.frame()[numeric(0),]
data.test[i,"Sample mean"] = mean.sample
data.test[i,"lower bound"] = left
data.test[i,"upper bound"] = right
data.test[i,"contains mean"] = 0
if (left < 0 && right > 0) {data.test[i,"contains mean"]=1}
data.test <- signif(data.test, digits = 4)

print.data <- function()
{
  print(data.test, row.names = FALSE)
  cat("\n")
}

print.data()
```

```
 Sample mean lower bound upper bound contains mean
    -0.04311     -0.1552     0.06896             1
```

```r
# for 1000 sample cases and 10 no. of trials

set.seed(1000)
no.of.trials <- 10
data <- data.frame()[numeric(0), ]
counter <- 0

for (i in 1:no.of.trials)
{
  # for a 95% confidence interval
  sample <- runif (1000, -1, 1)
  l.sample <- length(sample)
  mean.sample <- mean(sample)
  sd.sample <- sd(sample)

  error <- qt(0.975, df=l.sample-1)*sd.sample/sqrt(l.sample)
  left <- mean.sample - error
  right <- mean.sample + error
  data[i,"Sample mean"] = mean.sample
  data[i,"lower bound"] = left
  data[i,"upper bound"] = right
```

```
  if (left < 0 && right > 0)
  {
    data[i,"contains mean"]=1
    counter = counter + 1
  }
  else{data[i,"contains mean"]=0}
}

# finding success percentage and rounding data to significant digits
percent <- (counter/no.of.trials)*100
data <- signif(data, digits = 4)

# print.data function to print the required format
print.data <- function()
{
  cat("Number of trials: ", no.of.trials, "\n\n")
  print(data, row.names = FALSE)
  cat("\n")
  cat(percent, "percent of CI's contained the mean", "\n")
}

print.data()
```

```
Number of trials:  10

 Sample mean lower bound upper bound contains mean
  -0.0002918    -0.03585     0.03527             1
  -0.0217800    -0.05793     0.01438             1
  -0.0083480    -0.04398     0.02728             1
   0.0023420    -0.03301     0.03769             1
   0.0199200    -0.01543     0.05527             1
   0.0043810    -0.03140     0.04016             1
   0.0082400    -0.02814     0.04461             1
  -0.0191000    -0.05468     0.01649             1
  -0.0210500    -0.05674     0.01463             1
   0.0173800    -0.01903     0.05380             1

100 percent of CI's contained the mean
```

Question 4

15 points

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)

```
# setting up class kind and printing data for 1 record
medicalRecord <- function(x)
{
  class(x) <- "medicalRecord"
  names(x) <- c("name", "gender","date_of_birth", "date_of_admission",
                "pulse", "temperature", "fluid_intake")
  return(x)
}
set.seed(8)
medRecord <- makePatient()
medRecord <- medicalRecord(medRecord)

print(medRecord)
```

```
$name
[1] "Mev"

$gender
[1] male
Levels: female male

$date_of_birth
[1] "1976-08-09"

$date_of_admission
[1] "2011-03-14" "2013-10-30" "2013-02-27" "2012-08-23" "2011-11-16"

$pulse
[1] 67 81 95 74 81

$temperature
[1] 98.33 98.16 99.00 98.49 98.67

$fluid_intake
[1] 0.62 0.93 0.18 0.39 0.34

attr(,"class")
[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)

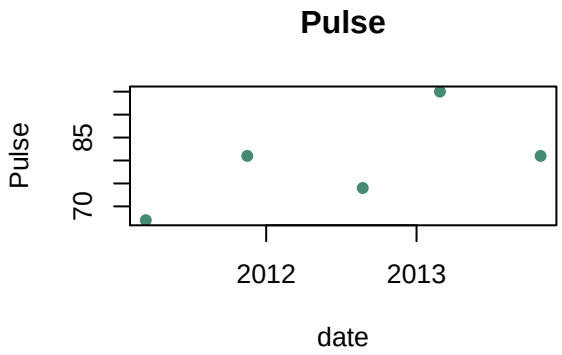
```
# defining mean method for class from generic mean
mean.medicalRecord <- function(x, class = "medicalRecord")
{
  pulse.mean = mean(x$pulse)
  temperature.mean = mean(x$temperature)
  fluids.mean = mean(x$fluid_intake)
  data.frame(cbind(pulse.mean, temperature.mean, fluids.mean))
}

# printing class type record as per requirements
print.medicalRecord <- function(x, class = "medicalRecord")
{
  name <- x$name
  gender <- x$gender
  date_of_birth = as.Date(x$date_of_birth, "%m/%d/%y")
  bdata = data.frame(date_of_birth)
  bdata = data.frame(cbind(name, gender, bdata))
  print(bdata, row.names = FALSE)
  cat("\n", "Medical record in chronological order", "\n\n")
  date_of_admission = as.Date(x$date_of_admission, "%m/%d/%y")
  data = data.frame(date_of_admission)
  pulse = c(x$pulse)
  temperature = c(x$temperature)
  fluid_intake = c(x$fluid_intake)
  data = data.frame(cbind(data, pulse, temperature, fluid_intake))
  data = data[order(as.Date(data$date, format="%d/%m/%Y")),]
  print(data, row.names = FALSE)
}

plot.medicalRecord <- function(x, class = "medicalrecord")
{
  par(mfrow=c(2,2))
  plot(x$date_of_admission, x$pulse,
       pch = 16, col = "aquamarine4",
       main="Pulse", xlab="date", ylab="Pulse")
  plot(x$date_of_admission, x$temperature,
       pch = 16, col = "chocolate",
       main="Temperature", xlab="date", ylab="Temperature")
  plot(x$date_of_admission, x$fluid_intake,
       pch = 16, col = "darkgoldenrod",
       main="Fluid_Intake", xlab="date", ylab="Fluid Intake")
}

# printing record's mean
mean.record <- mean.medicalRecord(medRecord)
print(mean.record, row.names = FALSE)
```

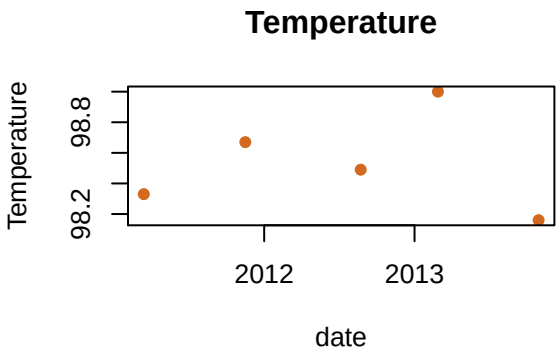
```
 pulse.mean temperature.mean fluids.mean
       79.6            98.53       0.492
```

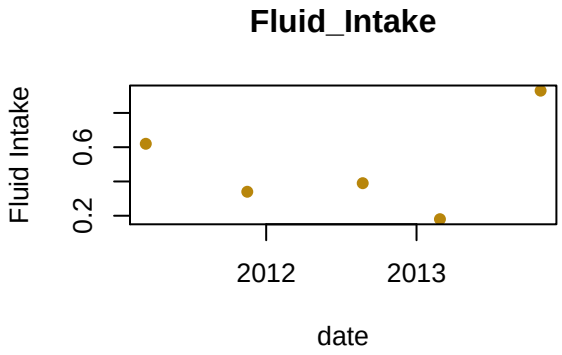
```
# printing medical record
print.medicalRecord(medRecord)
```

```
 name gender date_of_birth
  Mev   male    1976-08-09

 Medical record in chronological order

 date_of_admission pulse temperature fluid_intake
        2011-03-14    67       98.33         0.62
        2011-11-16    81       98.67         0.34
        2012-08-23    74       98.49         0.39
        2013-02-27    95       99.00         0.18
        2013-10-30    81       98.16         0.93
```

```
# plotting medical record data
plot.medicalRecord(medRecord)
```







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)

```
# setting up seed, class cohort and initializing 10 patients
cohort <- function(x)
{
  class(x) <- "cohort"
```

```
  names(x) <- c("name", "gender","date_of_birth", "date_of_admission",
                "pulse", "temperature", "fluid_intake")
  return(x)
}

set.seed(8)
n = 10
med.cohort = vector(mode = "list")
class(med.cohort) <- "cohort"
for (i in seq(1:n))
  {
    med.cohort[[i]] = makePatient()
    med.cohort[[i]] = cohort(med.cohort[[i]])
  }

# defining mean method for class from generic mean for class cohort
mean.cohort <- function(x, class = "cohort")
{
  # finding length of cohort list
  x.length = length(x)
  # initializing list
  pulse.mean = vector(mode = "list")
  temperature.mean = vector(mode = "list")
  fluids.mean = vector(mode = "list")
  # empty data frame
  colClasses = c("numeric","numeric", "numeric")
  col.names = c("pulse.mean","temp.mean","fluid.mean")
  mean.data <- read.table(text = "",
                colClasses = colClasses,
                col.names = col.names)
  # finding mean
  for (i in seq(1:x.length))
    {
      mean.data[i,"pulse.mean"] = mean(x[[i]]$pulse)
      mean.data[i,"temp.mean"] = mean(x[[i]]$temperature)
      mean.data[i,"fluid.mean"]  = mean(x[[i]]$fluid_intake)
    }
  # Appending patient no. in mean.data
  patient.no = c(seq(1:x.length))
  mean.data = cbind(patient.no, mean.data)

  # returning data frame
  return(mean.data)

}

# printing
# printing class type record as per requirements
print.cohort <- function(x, class = "cohort")
{
  # finding length of cohort list
  x.length = length(x)
  for (i in seq(1:x.length))
```

```
    {
      name <- x[[i]]$name
      gender <- x[[i]]$gender
      date_of_birth = as.Date(x[[i]]$date_of_birth, "%m/%d/%y")
      bdata = data.frame(date_of_birth)
      bdata = data.frame(cbind(name, gender, bdata))
      cat("\n\n", "For patient number: ", i, "\n")
      print(bdata, row.names = FALSE)
      cat("\n", "Medical record in chronological order", "\n\n")
      date_of_admission = as.Date(x[[i]]$date_of_admission, "%m/%d/%y")
      data = data.frame(date_of_admission)
      pulse = c(x[[i]]$pulse)
      temperature = c(x[[i]]$temperature)
      fluid_intake = c(x[[i]]$fluid_intake)
      data = data.frame(cbind(data, pulse, temperature, fluid_intake))
      data = data[order(as.Date(data$date, format="%d/%m/%Y")),]
      print(data, row.names = FALSE)
    }
}

# printing mean for pulse, temperature and fluid intake for all 10 patients
mean.data = mean.cohort(med.cohort)
print(mean.data, row.names = FALSE)
```

```
 patient.no pulse.mean temp.mean fluid.mean
          1      79.60     98.53     0.4920
          2      78.00     98.50     0.2450
          3      81.50     98.44     0.4033
          4      78.00     98.60     0.6500
          5      88.33     98.05     0.5867
          6      83.50     98.45     0.4525
          7      83.00     98.01     0.9700
          8      77.50     98.15     0.3367
          9      77.00     98.83     0.4450
         10      79.33     98.30     0.6583
```

```
# printing the entries of the data in a proper format
print.cohort(med.cohort)
```

```
 For patient number:  1
 name gender date_of_birth
  Mev   male    1976-08-09

 Medical record in chronological order

 date_of_admission pulse temperature fluid_intake
        2011-03-14    67       98.33         0.62
        2011-11-16    81       98.67         0.34
        2012-08-23    74       98.49         0.39
        2013-02-27    95       99.00         0.18
        2013-10-30    81       98.16         0.93
```

```
For patient number:  2
name gender date_of_birth
 Yul   male    1988-06-28

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2012-01-16    76       98.92         0.14
       2013-08-07    80       98.07         0.35


For patient number:  3
name gender date_of_birth
 Zet female    1970-06-13

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-03-21    79       98.58         0.22
       2010-04-01    73       98.32         0.61
       2012-08-29    88       98.47         0.59
       2013-06-01    84       98.22         0.25
       2013-11-03    72       98.54         0.03
       2014-02-05    93       98.51         0.72


For patient number:  4
name gender date_of_birth
 Qih female    1987-08-30

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2011-06-22    78        98.6         0.65


For patient number:  5
name gender date_of_birth
 Wut   male    1974-06-28

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-04-12    76       98.05         0.65
       2011-02-16    93       98.26         0.97
       2012-04-12    96       97.84         0.14


For patient number:  6
name gender date_of_birth
 Juy   male    1983-06-09
```

```
Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-03-10    81       99.11         0.66
       2010-03-25    90       98.58         0.26
       2010-04-18    75       98.58         0.60
       2010-06-10    88       97.53         0.29


For patient number:  7
name gender date_of_birth
 God female    1990-02-12

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-03-12    83       98.01         0.97


For patient number:  8
name gender date_of_birth
 Fut   male    1970-01-11

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2011-04-07    80       97.87         0.36
       2011-04-14    83       97.91         0.00
       2011-08-16    66       98.49         0.13
       2013-03-15    74       98.38         0.31
       2013-06-20    74       98.41         0.49
       2013-11-12    88       97.83         0.73


For patient number:  9
name gender date_of_birth
 Pet   male    1979-01-01

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-10-30    85       98.84         0.60
       2012-05-10    69       98.82         0.29


For patient number:  10
name gender date_of_birth
 Yed   male    1977-11-11

Medical record in chronological order

date_of_admission pulse temperature fluid_intake
       2010-01-28    63       97.95         0.94
       2010-03-06    81       98.45         0.67
```

```
2010-07-10    98       98.65         0.79
2010-08-27    66       97.68         0.36
2011-06-18    83       98.00         0.69
2013-01-06    85       99.07         0.50
```

Question 5

-5 bonus points

Use the simulated results from question 1 to create a three-dimensional pie chart (actually, don't).