Bios 6301: Assignment 6

Nick Strayer

23 November, 2015

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:

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

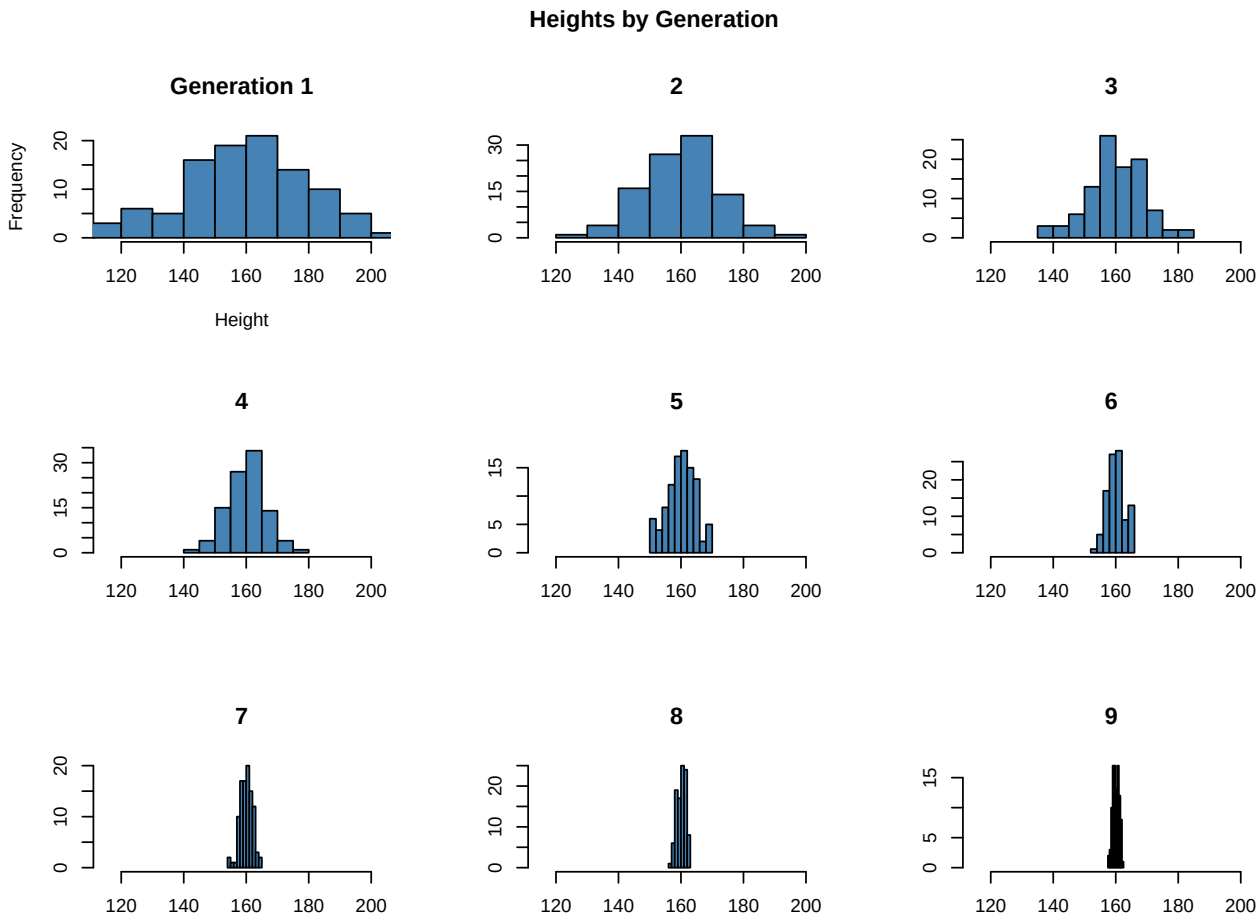
The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights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

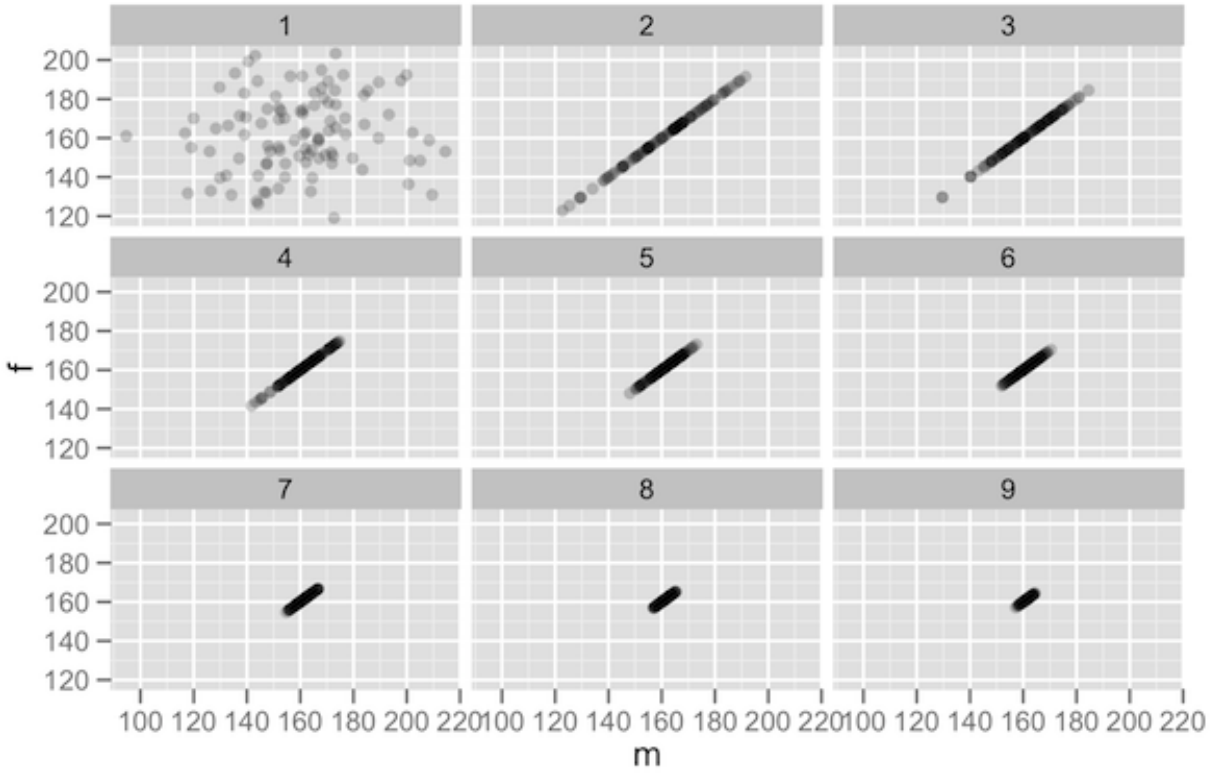
```
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
gens <- pop

firstMax <- max(pop$m) #grab values to use for range limits in plots.
firstMin <- min(pop$m)

par(mfrow = c(3,3)) #Set up a multiple plot environment
par(oma=c(1,1,2,1)) #Leave space for main title.

hist(pop$m, main = "Generation 1",   #histogram it.
             xlim = c(firstMin, firstMax),
             col  = "steelblue",
             xlab = "Height")
```

```
for(genNumber in 2:9){
  pop <- next_gen(pop)  #make the new population

  gens = rbind(gens, pop)

  hist(pop$m, main = as.character(genNumber),   #histogram it.
               xlim = c(firstMin, firstMax),
               col  = "steelblue",
               xlab = "", ylab = "")
}

title(main="Heights by Generation",outer=T)
```

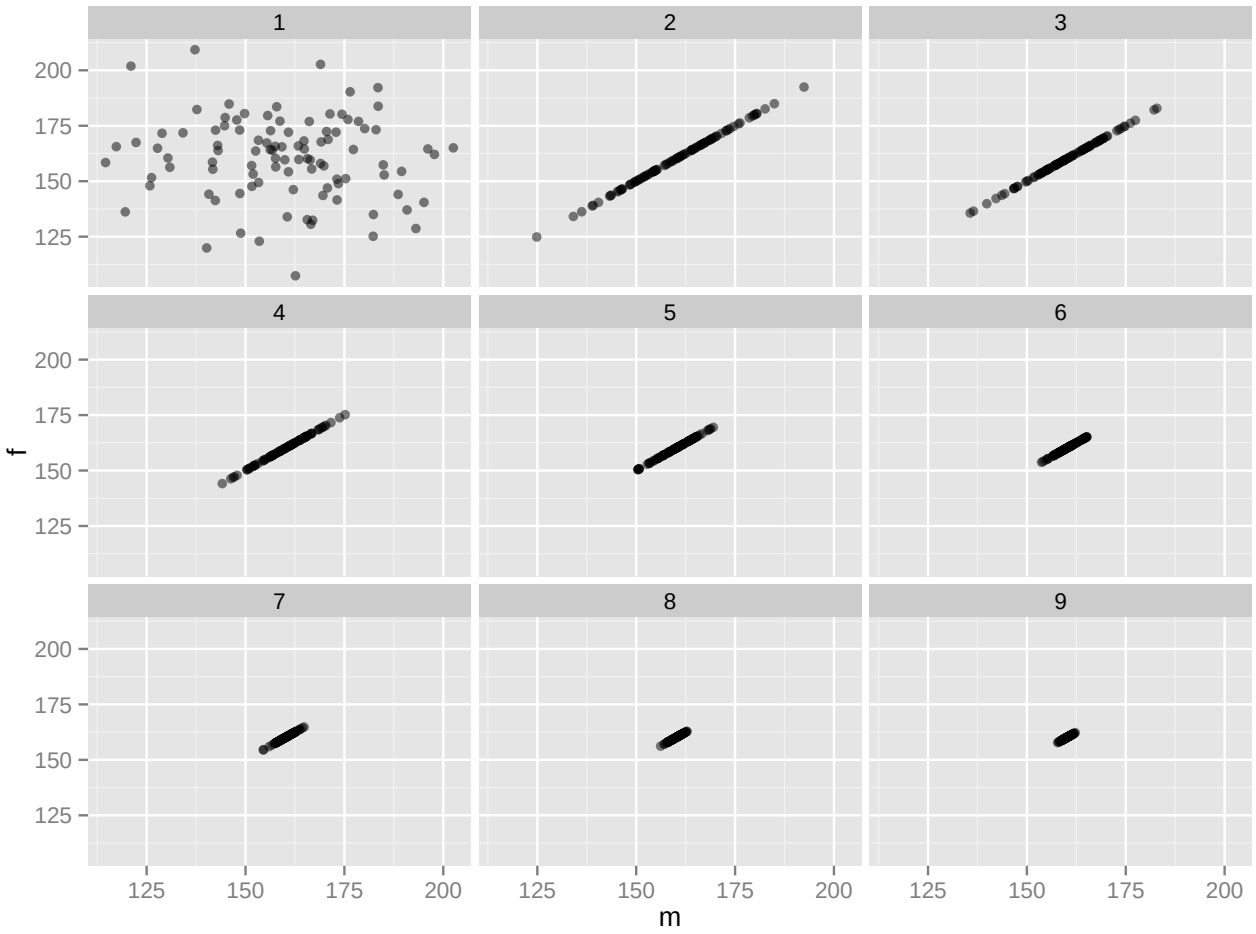


Question 2

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




```
gens$gen <- rep(1:9, each = 100) #give generation labels to the dataset
ggplot(gens, aes(x = m, y = f)) + geom_point(alpha = 0.5) + facet_wrap(~gen, nrow = 3)
```



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.

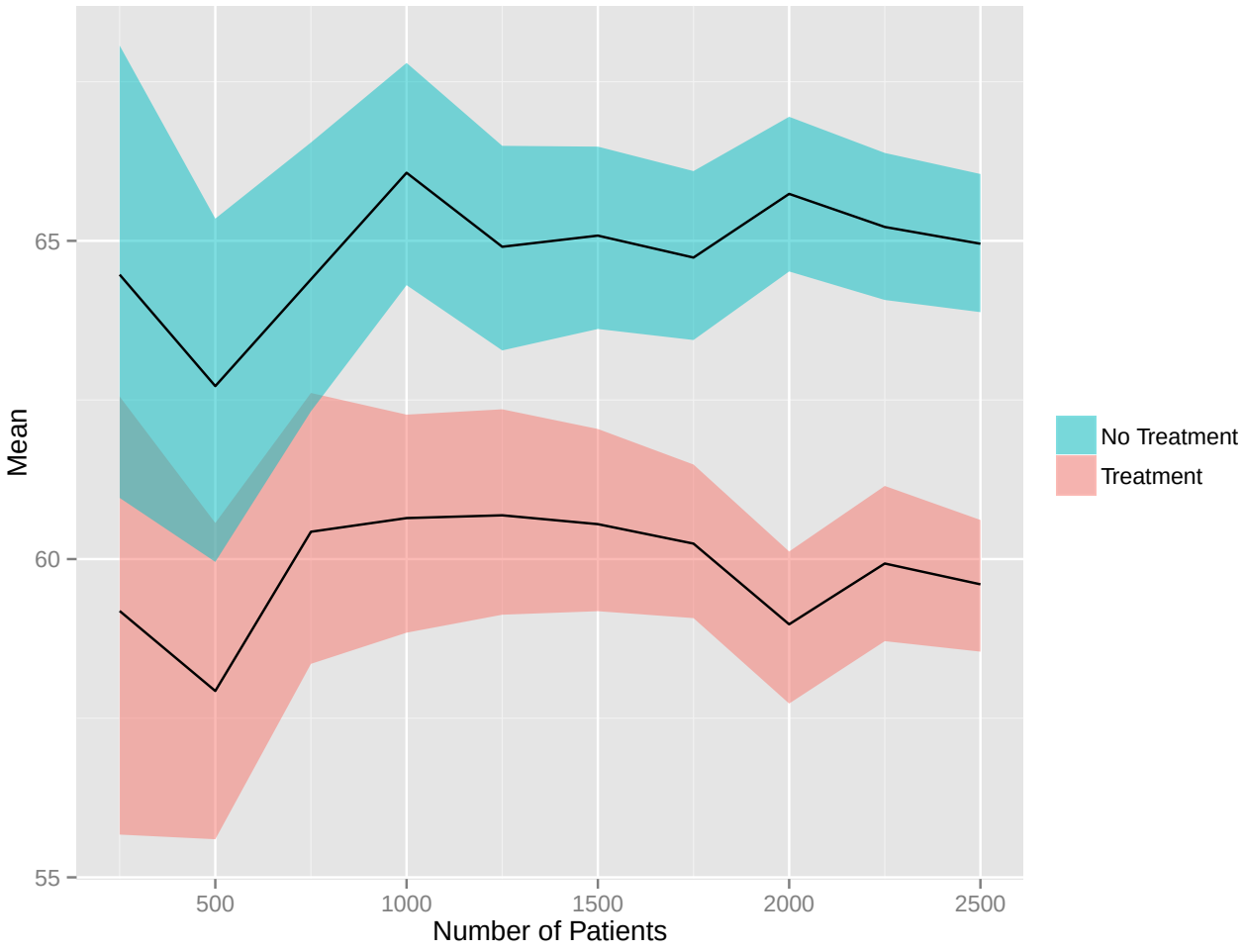
Starting with a sample size of 250, create a 95% bootstrap percentile interval for the mean of each group. Then create a new bootstrap interval by increasing the sample size by 250 until the sample is 2500. Thus you will create a total of 10 bootstrap intervals. Each bootstrap should create 1000 bootstrap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

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)

You may use base graphics or ggplot2. It should look similar to this (in base).

```
values = rnorm(10)
bootstrapMean = function(values){
  #Generate bootstraped medians
  bootstraped <- replicate(1000, mean(sample(values, length(values), replace=T)) )
  #Find the quantiles
  quantile(bootstraped, c(0.025, 0.975))
```

```
}

genVals <- function(num_patients = 250){
  #Set the treatment group options
  treatmentGroups <- c(0,1)

  #Set the treatments
  treatments <- sample(treatmentGroups, num_patients, replace = T)

  #Draw outcome values
  outcome_pre    <- rnorm(num_patients, mean = 60, sd = 20)

  #Adjust based on treatment.
  outcome_treat  <- ifelse(treatments==1, outcome_pre + 5, outcome_pre)

  #get seperate treat and non treat vectors and run the bootstrap function on them
  t <- outcome_treat[treatments == 1]
  n <- outcome_treat[treatments == 0]
  bsTreat    <- bootstrapMean(t)
  bsNonTreat <- bootstrapMean(n)


  nonTreatRes <- data.frame(lb = bsNonTreat[1], ub = bsNonTreat[2], m = mean(n),
                    treatment = 0,              n = num_patients)
  treatRes    <- data.frame(lb = bsTreat[1],    ub = bsTreat[2],    m = mean(t),
                    treatment = 1,              n = num_patients)
  #return a dataframe that has 4 columns, ub, lb, treatment (0,1), numPatients
  return(rbind(nonTreatRes,treatRes))
}

bsIntervals = genVals() #generate first interval.
for(i in 2:10){ bsIntervals = rbind(bsIntervals, genVals(250*i) ) }

#plot it!
ggplot(bsIntervals, aes(x = n, group = factor(treatment))) +
  geom_ribbon(aes(ymin = lb, ymax = ub, fill = factor(treatment)), alpha = 0.5) +
  geom_line(aes(y = m)) + labs("y" = "Mean", "x" = "Number of Patients") +
  scale_fill_discrete(guide = guide_legend(reverse=TRUE), name = "",
                      labels=c("Treatment","No Treatment"))
```



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 = name, gender = gender, date_of_birth = dob, date_of_admission = doa,
       pulse = pulse, temperature = temp, fluid_intake = fluid)
}
```

```
makePatient()
```

```
## $name
## [1] "Got"
##
## $gender
## [1] male
## Levels: female male
##
## $date_of_birth
## [1] "1971-04-24"
##
## $date_of_admission
## [1] "2010-06-24" "2010-03-27" "2012-11-21" "2010-09-10" "2010-01-15"
##
## $pulse
## [1] 81 71 62 81 78
##
## $temperature
## [1] 98.42 98.44 98.36 98.39 98.48
##
## $fluid_intake
## [1] 0.58 0.63 0.34 0.39 0.17
```

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)

```
set.seed(8)
p <- makePatient()
class(p) <- 'medicalRecord'
p
```

```
## $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)

First we write the class functions

```
mean.medicalRecord <- function(d){
  return(list(pulseAvg = mean(d$pulse), tempAvg = mean(d$temperature),
              fluidsAvg = mean(d$fluid_intake)))
}

print.medicalRecord <- function(d){
  cat(paste("Name:", d$name, "\n"))
  cat("The patient is a", d$gender, "and was born on", d$date_of_birth, "\n")
  cat("Visits:\n")
  #make a df of visits to ease ordering by date.
  visits = data.frame(doa = d$date_of_admission, p = d$pulse, t = d$temperature, fi = d$fluid_intake)
  visits = visits[order(visits$doa), ]
  cat(sprintf("Date: %s | Pulse: %3.2s | Temp: %6.4s | Fluid Intake: %6.4s\n",
    visits$doa, visits$p, visits$t, visits$fi))
}

plot.medicalRecord <- function(d){
  visits = data.frame(doa = d$date_of_admission, p = d$pulse, t = d$temperature, fi = d$fluid_intake)
  visits = visits[order(visits$doa), ]
  par(mfrow = c(3,1))
  plot(visits$doa, visits$p, type = "l", xlab = "", ylab = "", main = "Pulse")
  plot(visits$doa, visits$t, type = "l", xlab = "", ylab = "value", main = "Temperature")
  plot(visits$doa, visits$fi, type = "l", xlab = "Date", ylab = "", main = "Fluid Levels")
}
```

. . . and then we call them.

First `mean`:

```
mean(p)
```

```
## $pulseAvg
## [1] 79.6
##
## $tempAvg
## [1] 98.53
##
## $fluidsAvg
## [1] 0.492
```

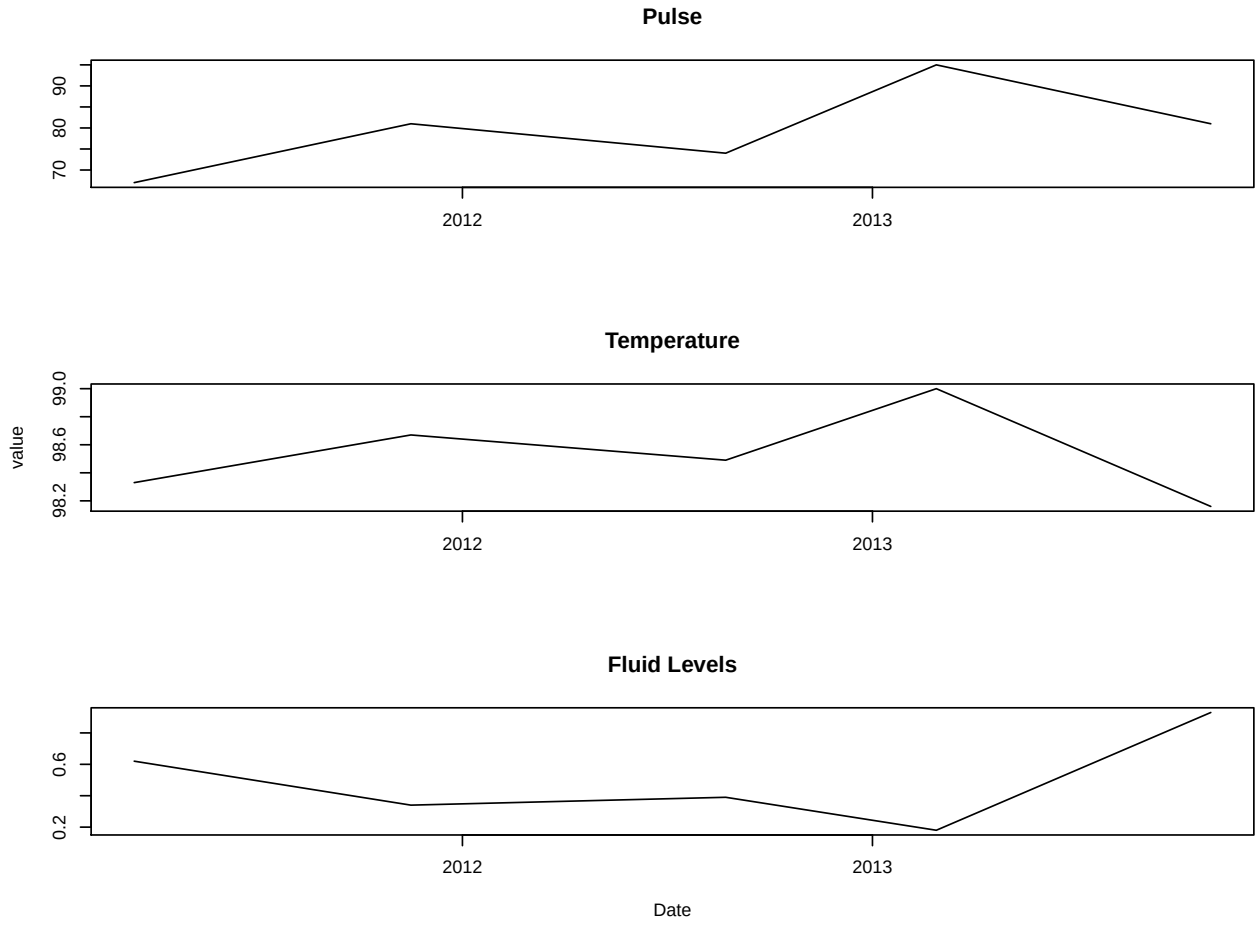
Second `print`:

```
p
```

```
## Name: Mev
## The patient is a 2 and was born on 2412
## Visits:
## Date: 2011-03-14 | Pulse:  67 | Temp:   98.3 | Fluid Intake:   0.62
##  Date: 2011-11-16 | Pulse:  81 | Temp:   98.6 | Fluid Intake:   0.34
##  Date: 2012-08-23 | Pulse:  74 | Temp:   98.4 | Fluid Intake:   0.39
##  Date: 2013-02-27 | Pulse:  95 | Temp:     99 | Fluid Intake:   0.18
##  Date: 2013-10-30 | Pulse:  81 | Temp:   98.1 | Fluid Intake:   0.93
```

Last `plot`:

```
plot(p)
```



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)

```
j <- list(name="Joe", salary=55000, union=TRUE)
class(j) <- 'employee'
attributes(j)
```

```
## $names
## [1] "name"   "salary" "union"
##
## $class
## [1] "employee"
```

```
print.employee <- function(wrkr) {
  cat(sprintf("name: %s\nsalary: %s\nunion member: %s",
              wrkr$name, wrkr$salary, wrkr$union), "\n")
}

j
```

```
## name: Joe
## salary: 55000
## union member: TRUE
```