Bios 6301: Assignment 6

Yudong Cao

Due Thursday, 1 December, 1:00 PM

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

50 points total.

Grade: 46/50

Submit a single knitr file (named `homework6.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 `homework6.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))
```

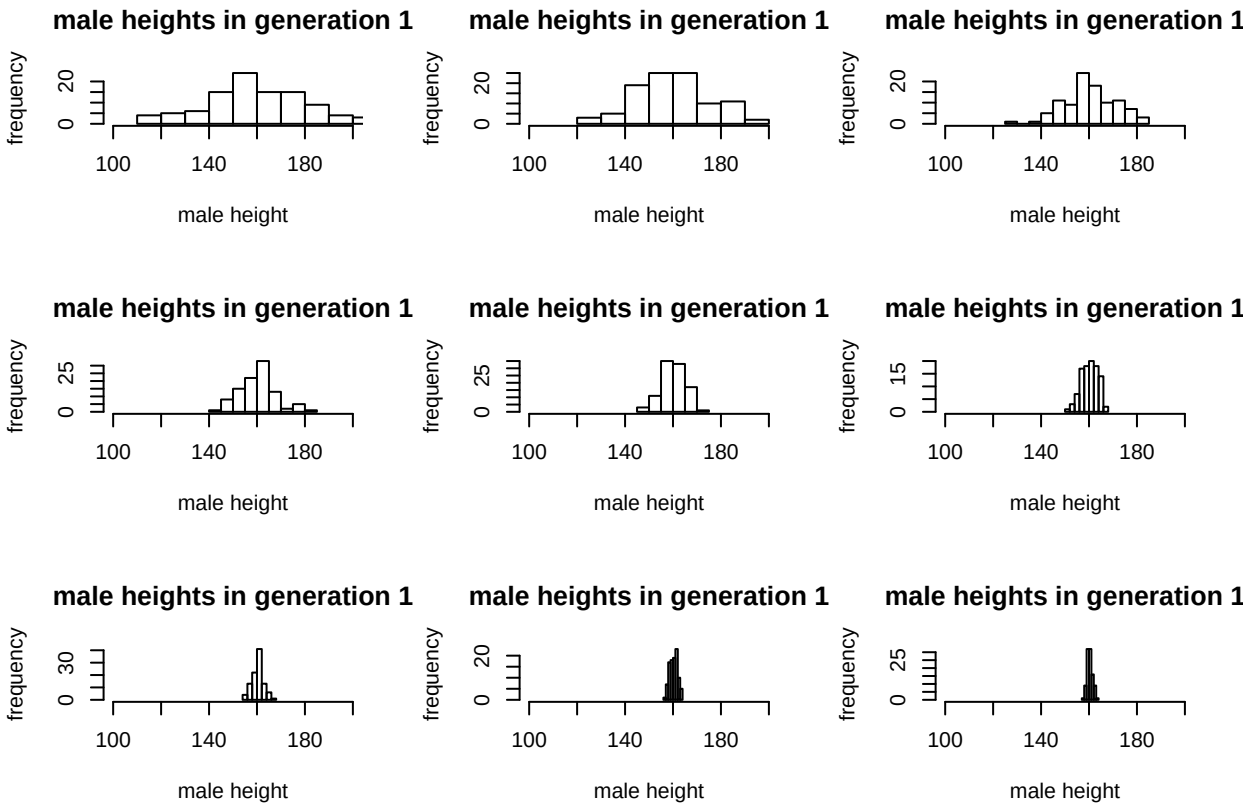
The following function takes the data frame `pop` and randomly permutes the ordering of the men. Men and women are then paired according to rows, and heights for the next generation are calculated by taking the mean of each row. The function returns a data frame with the same structure, giving the heights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

```
par(mfrow=c(3,3))
gen_1 <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
hist(gen_1$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_2 <- next_gen(gen_1)
hist(gen_2$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_3 <- next_gen(gen_2)
```

```r
hist(gen_3$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_4 <- next_gen(gen_3)
hist(gen_4$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_5 <- next_gen(gen_4)
hist(gen_5$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_6 <- next_gen(gen_5)
hist(gen_6$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_7 <- next_gen(gen_6)
hist(gen_7$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_8 <- next_gen(gen_7)
hist(gen_8$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
gen_9 <- next_gen(gen_8)
hist(gen_9$m,main=paste("male heights in generation 1"),xlim=c(100,200),xlab="male height",ylab="freque
```



Question 2

10 points

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

```r
library(ggplot2)
gen <- rbind(gen_1,gen_2,gen_3,gen_4,gen_5,gen_6,gen_7,gen_8,gen_9)
gennb <- c(rep(1:9,each=100))
gen <- cbind(gen,gennb)
ggplot(gen) + geom_point(mapping=aes(m,f),alpha=1/7) + facet_wrap(~gennb)
```

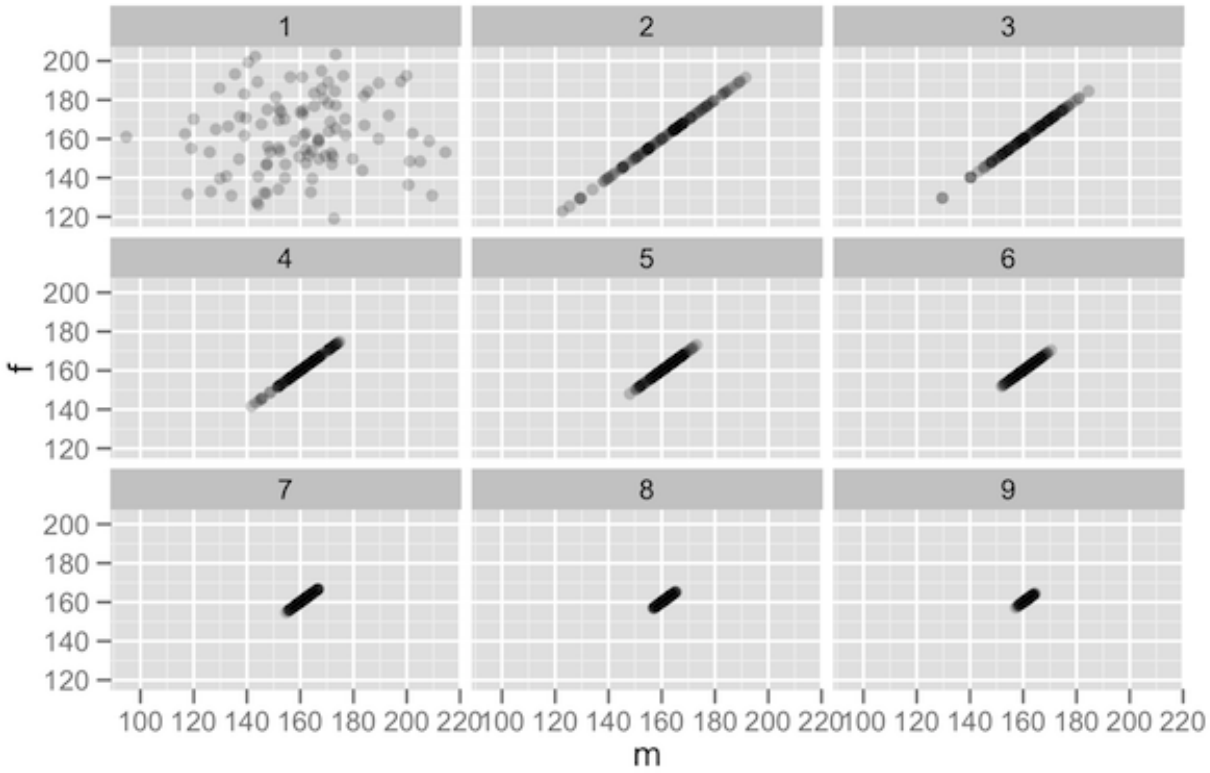
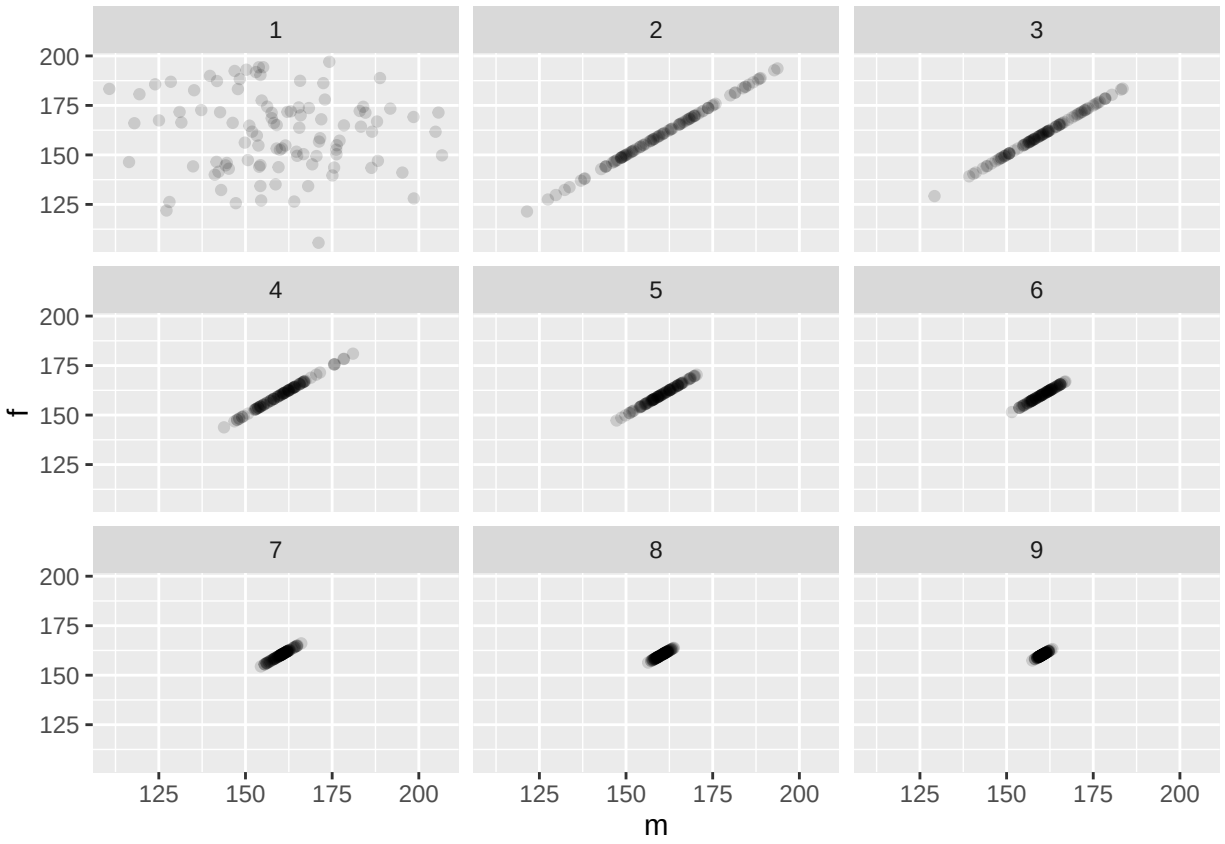


Figure 1: generations plot



Question 3

10 points

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)

```
df<-data.frame(numeric(0),numeric(0),numeric(0),numeric(0),numeric(0),numeric(0),numeric(0))
for(n in seq(250,2500,by=250)) {
  trt<-rbinom(n,1,.5)
  out<-rnorm(n,60,20)+5*trt
  dat<-data.frame(trt,out)
  m<-1000
  bootstrap<-data.frame(numeric(m),numeric(m))
  for (j in 1:m) {
    bootstrap[j,1]<-mean(sample(dat$out[trt==0],n,replace=T))
    bootstrap[j,2]<-mean(sample(dat$out[trt==1],n,replace=T))
  }
  q0<-quantile(bootstrap[,1],probs=c(.025,.5,.975))
  q1<-quantile(bootstrap[,2],probs=c(.025,.5,.975))
  qt<-c(n,q0,q1)
```

```
  df<-rbind(df,qt)
}
colnames(df)<-c('n','2.5% ctrl','50% ctrl','97.5% ctrl','2.5% trt','50% trt','97.5% trt')
```

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).

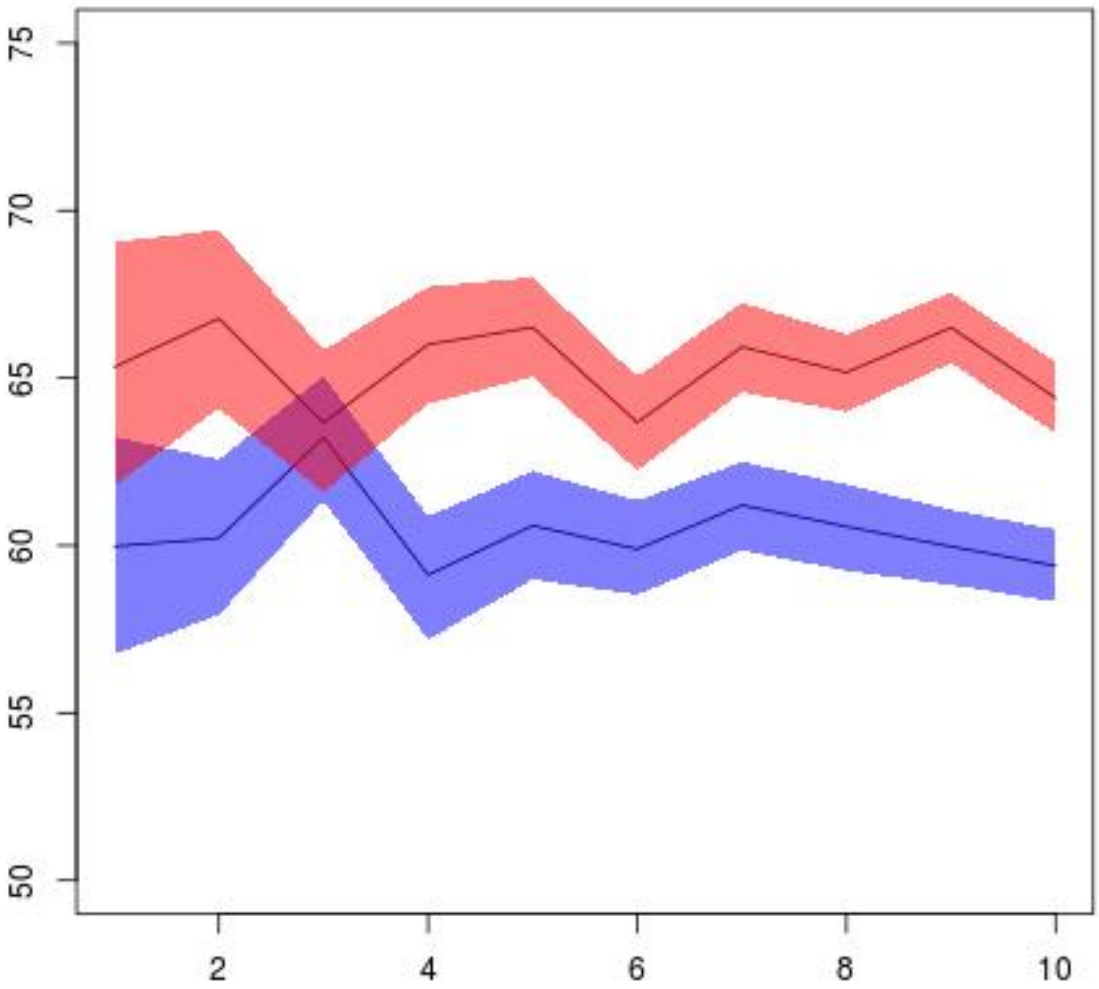
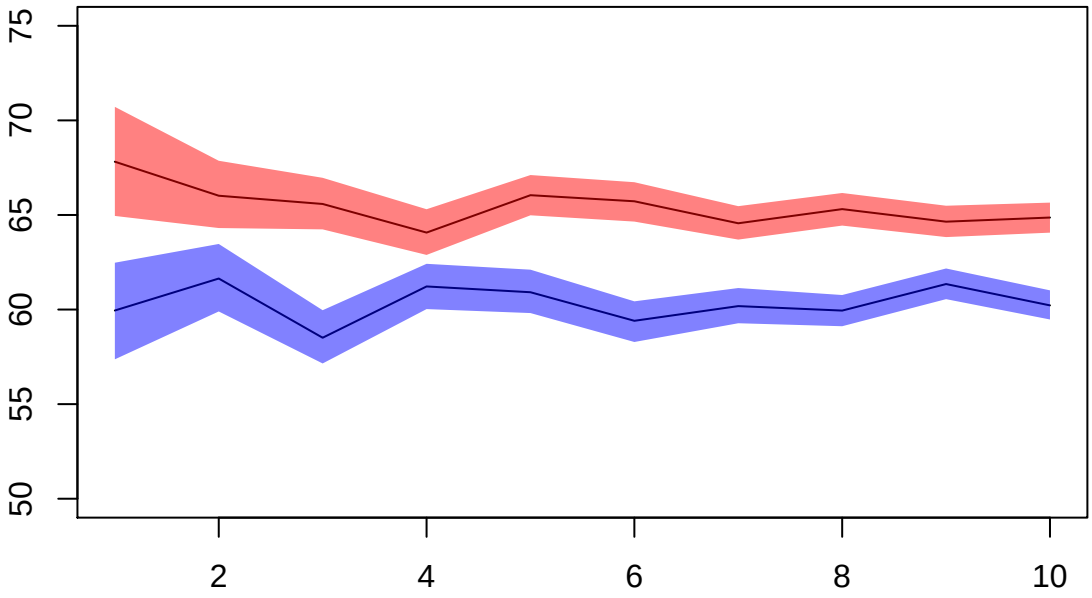


Figure 2: bp interval plot

```
makeTransparent = function(..., alpha=0.5) {
  if(alpha<0 | alpha>1) stop("alpha must be between 0 and 1")
  alpha = floor(255*alpha)
```

```
  newColor = col2rgb(col=unlist(list(...)), alpha=FALSE)
  .makeTransparent = function(col, alpha) {
    rgb(red=col[1], green=col[2], blue=col[3], alpha=alpha, maxColorValue=255)
  }
  newColor = apply(newColor, 2, .makeTransparent, alpha=alpha)
  return(newColor)
}

par(new=FALSE)
plot(NULL,
  xlim=c(1, 10),
  ylim=c(50, 75),
  xlab="",
  ylab=""
)
lines(df[,'n']/250,df[,'50% ctrl'])
lines(df[,'n']/250,df[,'50% trt'])
polygon(x=c(rev(df[,'n']/250),df[,'n']/250),
        y=c(rev(df[,'97.5% ctrl']),df[,'2.5% ctrl']), border=NA, col=makeTransparent('blue',alpha=0.5))
polygon(x=c(rev(df[,'n']/250),df[,'n']/250),
        y=c(rev(df[,'97.5% trt']),df[,'2.5% trt']), border=NA, col=makeTransparent('red',alpha=0.5))
```

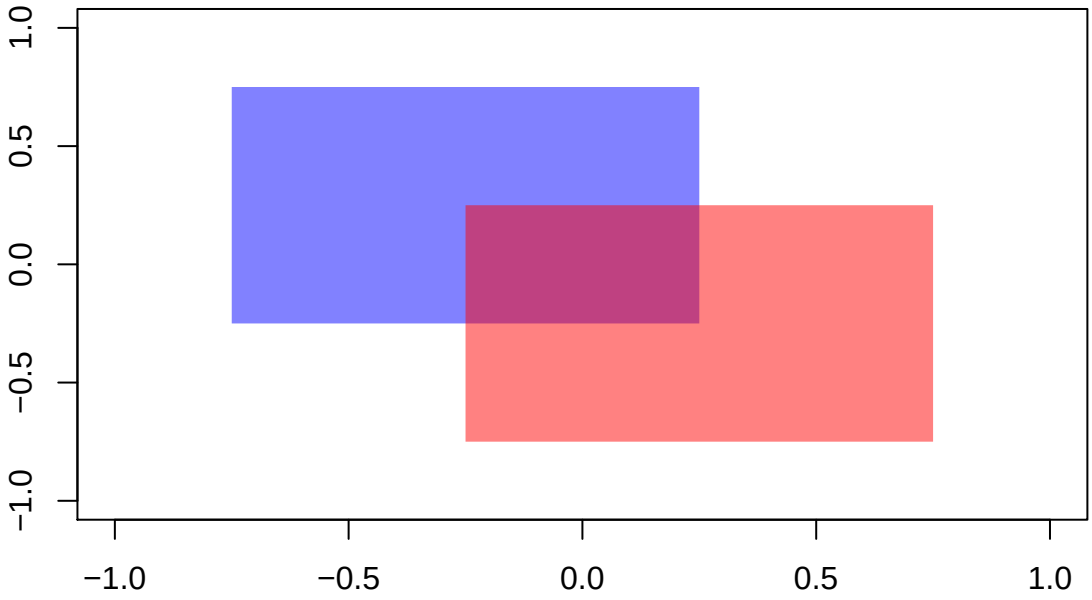


Here's an example of how you could create transparent shaded areas.

```
makeTransparent = function(..., alpha=0.5) {
  if(alpha<0 | alpha>1) stop("alpha must be between 0 and 1")
  alpha = floor(255*alpha)
  newColor = col2rgb(col=unlist(list(...)), alpha=FALSE)
  .makeTransparent = function(col, alpha) {
    rgb(red=col[1], green=col[2], blue=col[3], alpha=alpha, maxColorValue=255)
  }
  newColor = apply(newColor, 2, .makeTransparent, alpha=alpha)
  return(newColor)
}
```

```r
par(new=FALSE)
plot(NULL,
  xlim=c(-1, 1),
  ylim=c(-1, 1),
  xlab="",
  ylab=""
)

polygon(x=c(seq(-0.75, 0.25, length.out=100), seq(0.25, -0.75, length.out=100)),
        y=c(rep(-0.25, 100), rep(0.75, 100)), border=NA, col=makeTransparent('blue',alpha=0.5))
polygon(x=c(seq(-0.25, 0.75, length.out=100), seq(0.75, -0.25, length.out=100)),
        y=c(rep(-0.75, 100), rep(0.25, 100)), border=NA, col=makeTransparent('red',alpha=0.5))
```



Question 4

15 points

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

```r
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)

```
set.seed(8)
c<-makePatient()
names(c)<-list('name', 'gender', 'date_of_birth', 'date_of_admission', 'pulse', 'temperature', 'fluid_i
class(c)<-'medicalRecord'
print(c)
```

```
## $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"
```

```
attributes(c)
```

```
## $names
## [1] "name"              "gender"            "date_of_birth"
## [4] "date_of_admission" "pulse"             "temperature"
## [7] "fluid_intake"
##
## $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)

```
mean.medicalRecord<-function(x) {
  mean.pulse<-mean(x$pulse)
  mean.temperature<-mean(x$temperature)
  mean.fluids<-mean(x$fluid_intake)
  return(c(mean.pulse,mean.temperature,mean.fluids))
}
```

```
mean.medicalRecord(c)
```

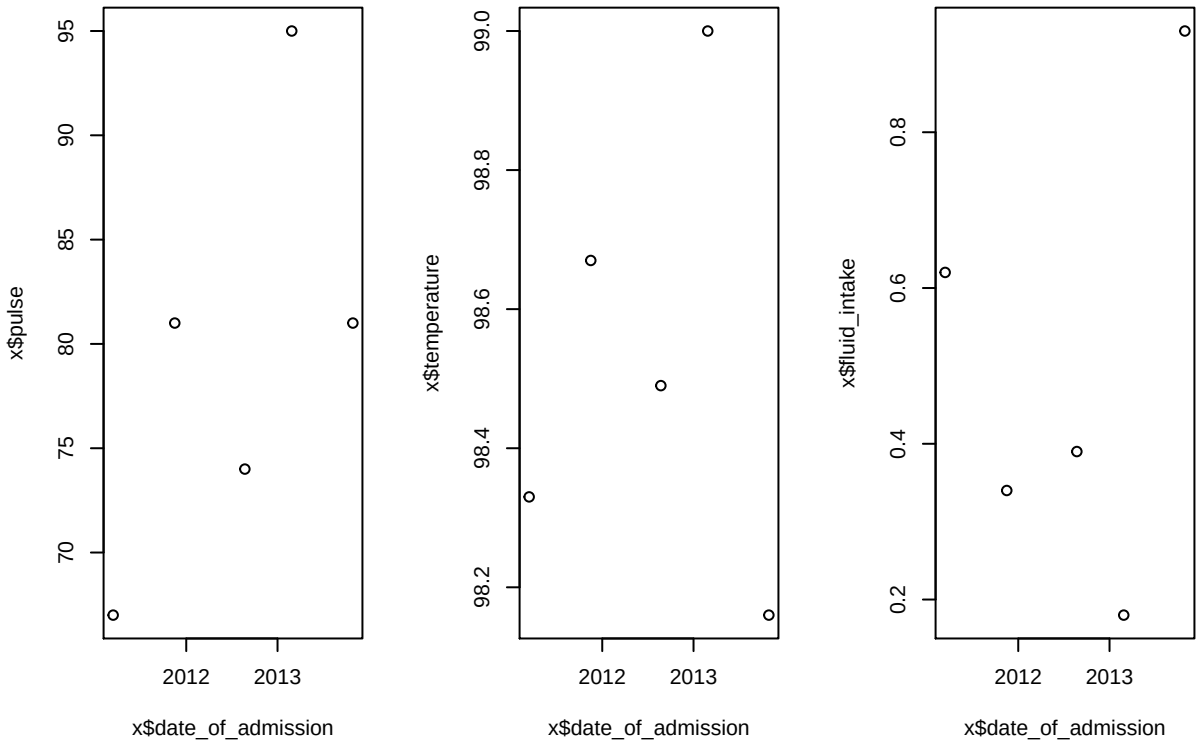
```
## [1] 79.600 98.530  0.492
```

```
print.medicalRecord<-function(x) {
  cat(sprintf("name: %s\ngender: %s\ndate of birth: %s",
              x$name, x$gender, x$date_of_birth), "\n")
  date<-as.data.frame(x[4:7])
  date.order<-date[order(date[,1]),]
  return(date.order)
}
print.medicalRecord(c)
```

```
## name: Mev
## gender: male
## date of birth: 1976-08-09
```

```
##   date_of_admission pulse temperature fluid_intake
## 1        2011-03-14    67       98.33         0.62
## 5        2011-11-16    81       98.67         0.34
## 4        2012-08-23    74       98.49         0.39
## 3        2013-02-27    95       99.00         0.18
## 2        2013-10-30    81       98.16         0.93
```

```
plot.medicalRecord<-function(x) {
  par(mfrow=c(1,3))
  plot(x$date_of_admission,x$pulse)
  plot(x$date_of_admission,x$temperature)
  plot(x$date_of_admission,x$fluid_intake)
}
plot.medicalRecord(c)
```



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)
mean.medicalRecord<-function(x) {
  mean.pulse<-mean(x$pulse)
  mean.temperature<-mean(x$temperature)
  mean.fluids<-mean(x$fluid_intake)
  return(c(mean.pulse,mean.temperature,mean.fluids))
}
for (i in 1:10) {
  cohort_i<-makePatient()
  class(cohort_i)<-'medicalRecord'
  names(cohort_i)<-list('name', 'gender', 'date_of_birth', 'date_of_admission', 'pulse', 'temperature',
  print(print(cohort_i))
  print(c('mean:',mean.medicalRecord(cohort_i)))
}
```

```
## name: Mev
## gender: male
## date of birth: 1976-08-09
##   date_of_admission pulse temperature fluid_intake
## 1        2011-03-14    67       98.33         0.62
## 5        2011-11-16    81       98.67         0.34
## 4        2012-08-23    74       98.49         0.39
## 3        2013-02-27    95       99.00         0.18
## 2        2013-10-30    81       98.16         0.93
## [1] "mean:" "79.6"  "98.53" "0.492"
## name: Yul
## gender: male
## date of birth: 1988-06-28
##   date_of_admission pulse temperature fluid_intake
## 1        2012-01-16    76       98.92         0.14
## 2        2013-08-07    80       98.07         0.35
## [1] "mean:"  "78"     "98.495" "0.245"
## name: Zet
## gender: female
## date of birth: 1970-06-13
##   date_of_admission pulse temperature fluid_intake
## 6        2010-03-21    79       98.58         0.22
## 5        2010-04-01    73       98.32         0.61
## 4        2012-08-29    88       98.47         0.59
## 3        2013-06-01    84       98.22         0.25
## 1        2013-11-03    72       98.54         0.03
## 2        2014-02-05    93       98.51         0.72
## [1] "mean:"             "81.5"              "98.44"
## [4] "0.403333333333333"
## name: Qih
## gender: female
## date of birth: 1987-08-30
##   date_of_admission pulse temperature fluid_intake
## 1        2011-06-22    78        98.6         0.65
## [1] "mean:" "78"    "98.6"  "0.65"
```

```
## name: Wut
## gender: male
## date of birth: 1974-06-28
##   date_of_admission pulse temperature fluid_intake
## 3        2010-04-12    76       98.05         0.65
## 1        2011-02-16    93       98.26         0.97
## 2        2012-04-12    96       97.84         0.14
## [1] "mean:"             "88.3333333333333"  "98.05"
## [4] "0.586666666666667"
## name: Juy
## gender: male
## date of birth: 1983-06-09
##   date_of_admission pulse temperature fluid_intake
## 4        2010-03-10    81       99.11         0.66
## 1        2010-03-25    90       98.58         0.26
## 3        2010-04-18    75       98.58         0.60
## 2        2010-06-10    88       97.53         0.29
## [1] "mean:"  "83.5"   "98.45"  "0.4525"
## name: God
## gender: female
## date of birth: 1990-02-12
##   date_of_admission pulse temperature fluid_intake
## 1        2010-03-12    83       98.01         0.97
## [1] "mean:" "83"    "98.01" "0.97"
## name: Fut
## gender: male
## date of birth: 1970-01-11
##   date_of_admission pulse temperature fluid_intake
## 5        2011-04-07    80       97.87         0.36
## 4        2011-04-14    83       97.91         0.00
## 2        2011-08-16    66       98.49         0.13
## 1        2013-03-15    74       98.38         0.31
## 6        2013-06-20    74       98.41         0.49
## 3        2013-11-12    88       97.83         0.73
## [1] "mean:"             "77.5"              "98.1483333333333"
## [4] "0.336666666666667"
## name: Pet
## gender: male
## date of birth: 1979-01-01
##   date_of_admission pulse temperature fluid_intake
## 1        2010-10-30    85       98.84         0.60
## 2        2012-05-10    69       98.82         0.29
## [1] "mean:" "77"    "98.83" "0.445"
## name: Yed
## gender: male
## date of birth: 1977-11-11
##   date_of_admission pulse temperature fluid_intake
## 4        2010-01-28    63       97.95         0.94
## 3        2010-03-06    81       98.45         0.67
## 1        2010-07-10    98       98.65         0.79
## 6        2010-08-27    66       97.68         0.36
## 5        2011-06-18    83       98.00         0.69
## 2        2013-01-06    85       99.07         0.50
## [1] "mean:"             "79.3333333333333"  "98.3"
```

```
## [4] "0.658333333333333"
```

JC Grading -4 To create a class, first create a function that returns an object of class cohort. Then develop methods mean.cohort and print.cohort. Check out Cole's solution on slack.