Bios 6301: Assignment 6

Hannah Weeks

Due Thursday, 3 December, 1:00 PM

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

50 points total.

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.

```r
library(sfsmisc)
library(ggplot2)
```

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:

```r
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.

```r
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.

Solution
In order to standardize the graphs to make comparisons easier, we use the most extreme values of `pop` as the x-axis limits. Since the `next_gen` function samples from the current generation, future generations should not have values which fall outside these limits.

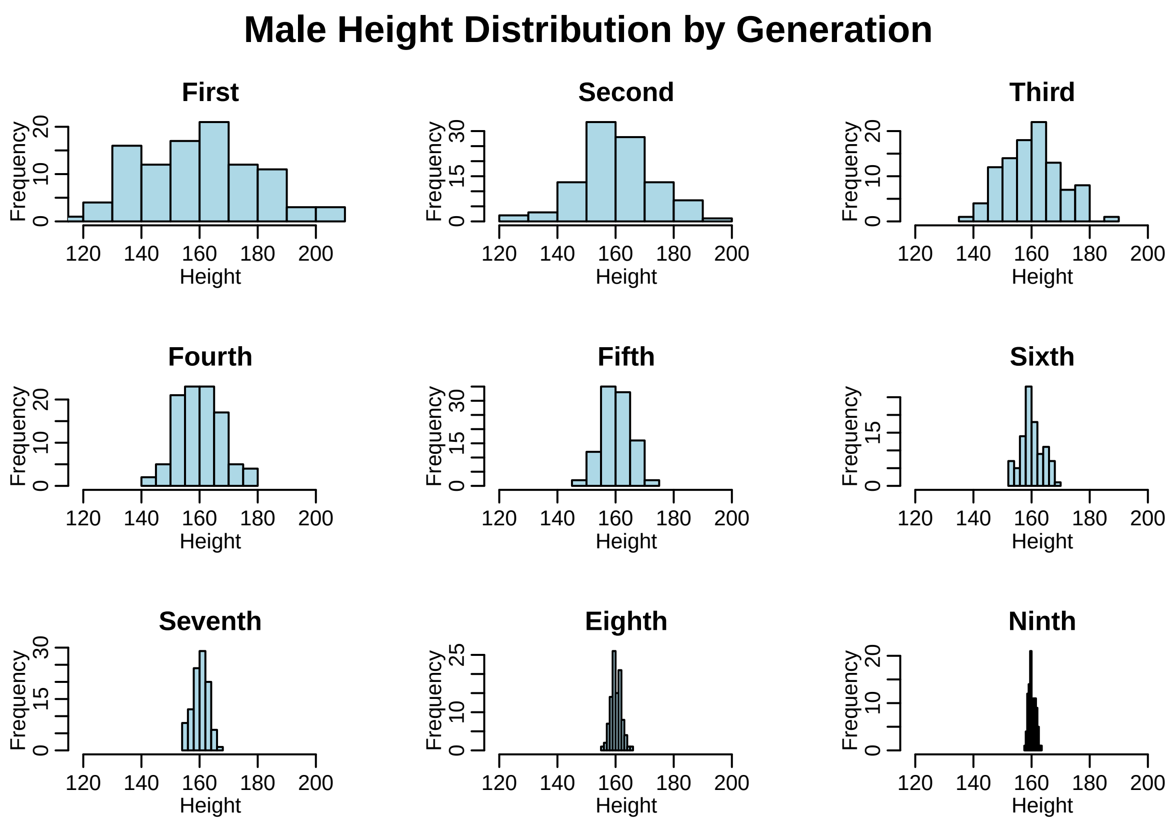
```r
#x-axis limits to make graphs consistent
xmin <- min(pop$m)
xmax <- max(pop$m)
#Labels for graphs corresponding to each generation
ordinal <- c("First", "Second", "Third", "Fourth", "Fifth", "Sixth", "Seventh", "Eighth", "Ninth")
```

Now, using `next_gen`, we create the height data for nine generations. Additionally, we add a column that indicates the generation to which that height value belongs. Using the `sfsmisc` package, we plot all nine histograms together.

```r
#Generate height data for each generation
genData <- pop
genData$gen <- 1
for(n in 2:9){
  #Ignore generation label when evalutating next generation heights
  pop <- next_gen(pop[,1:2])
  #Label generation number
  pop$gen <- n
  genData <- rbind(genData,pop)
}
tail(genData)
```

```
##            m        f gen
## 895 159.9877 159.9877   9
## 896 158.8811 158.8811   9
## 897 159.9080 159.9080   9
## 898 162.9869 162.9869   9
## 899 161.0519 161.0519   9
## 900 160.7759 160.7759   9
```

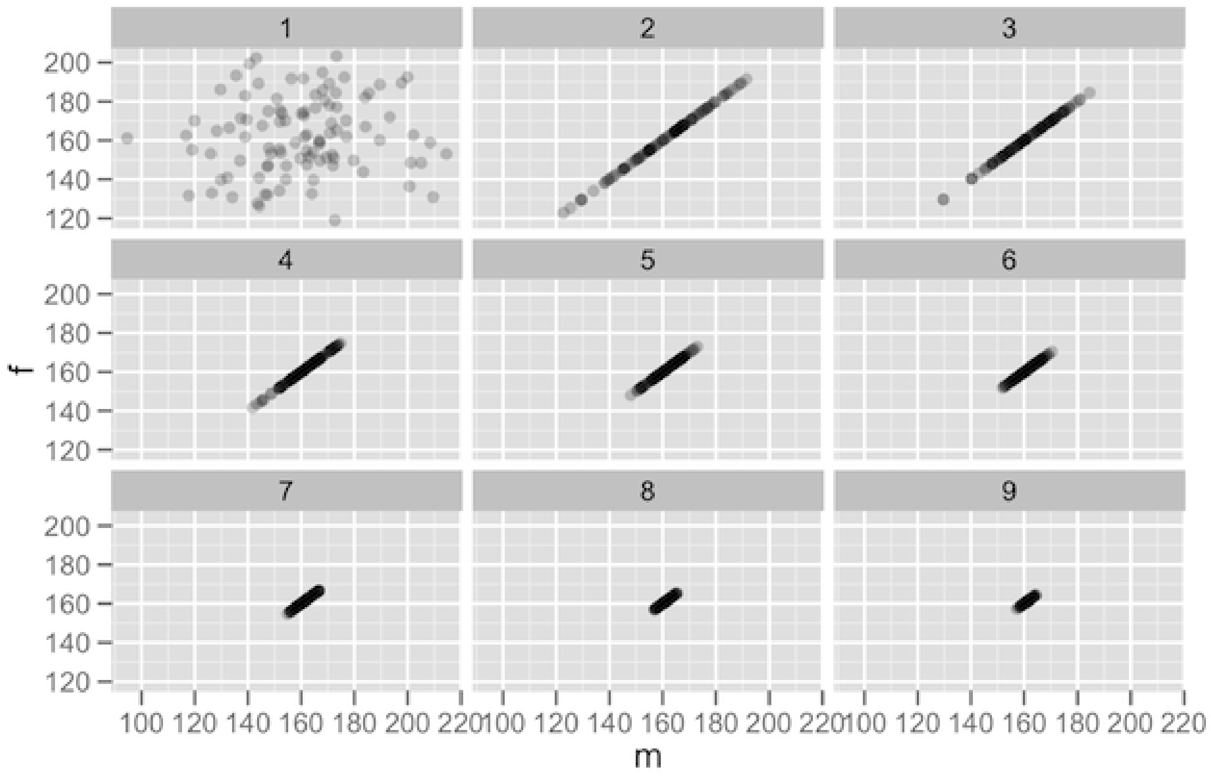
```r
#Setup for plotting 9 histograms together
mult.fig(9, main = "Male Height Distribution by Generation")
for(n in 1:9){
  #Subset out the data for that generation
  gen <- genData[genData$gen == n,]
  #Plot that generation's data
  hist(gen$m, main=ordinal[n], xlab="Height", col="light blue", xlim=c(xmin,xmax))
}
```



Question 2

10 points

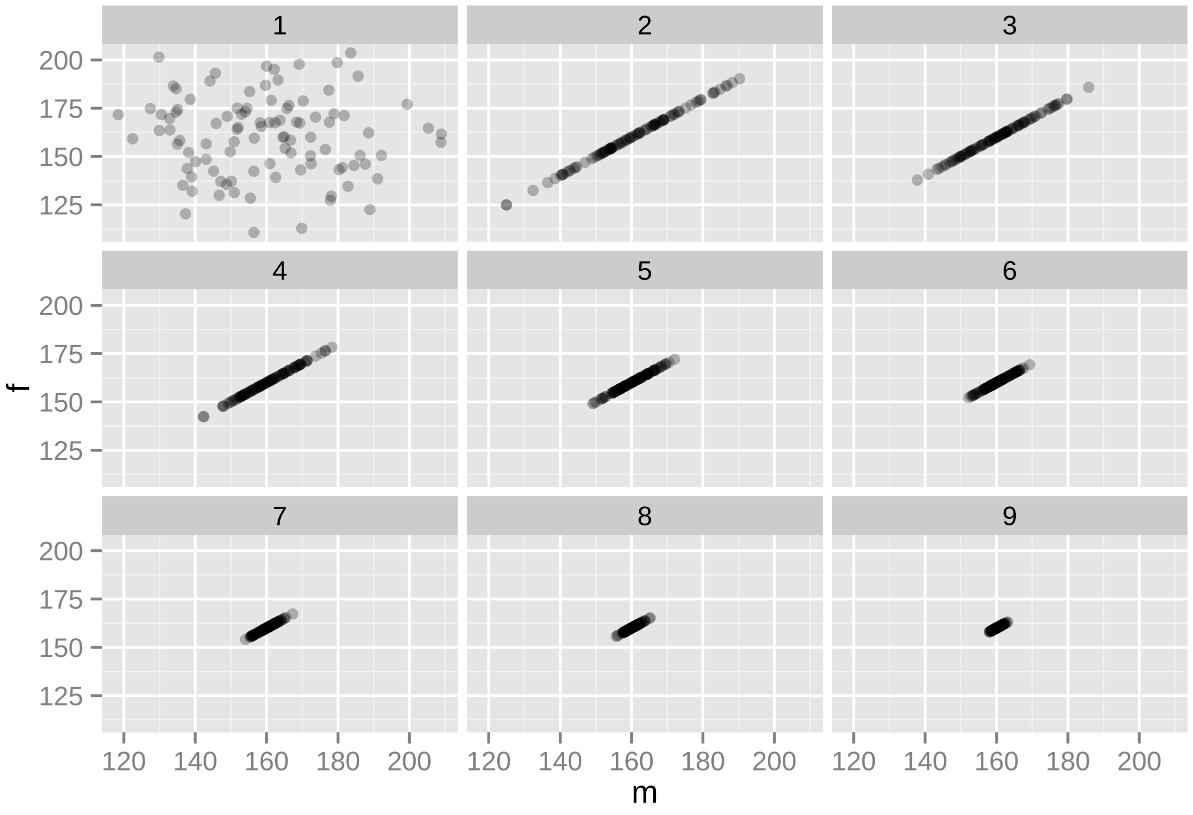
Use the simulated results from question 1 to reproduce (as closely as possible) the following plot in ggplot2.



Solution

Using the `qplot` function of `ggplot2`:
From `genData`, each plot is comparing the male data (`m`) with the female data (`f`). By specifying both `x` and `y` arguments, `qplot` will default to creating scatterplots. We facet on generation to create nine separate plots. The `alpha` argument is used to set the transparency of each data point.

```
qplot(x = m, y = f, data = genData, facets = ~gen, alpha = I(0.25))
```



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)

Solution
We write a function (with setup pulled from my function used in Assignment 2) that will calculate the 95% Bootstrap Percentile Confidence Interval. After creating the outcome effects as specified (i.e. adding 5 for treatment group), the data is separated based on group. For each treatment group, we calculate the mean of 1,000 bootstrapped samples and use `quantile(..., probs = c(.025,.975))` to obtain the 95% CI.

```
#Set default arguments:
#sampleSize = number of patients
#nboot = number of bootstrap samples
#alpha = significance level
bootPercentCI <- function(sampleSize, nboot = 1000, alpha = .05){
  #Assign subjects to treatment (1) or control (0)
  group <- rbinom(n = sampleSize, size = 1, prob = .5)
  outcome <- rnorm(n = sampleSize, mean = 60, sd = 20)
  treatmentOutcome <- outcome + 5
  #If subject in treatment group, update outcome to include treatment effect
  outcomeEffects <- treatmentOutcome*group + outcome*(1-group)
  data <- data.frame(group,outcomeEffects)

  #Separate effects based on treatment group
  t0 <- data[data$group == 0,]$outcomeEffects
  t1 <- data[data$group == 1,]$outcomeEffects

  #Perform bootstrap with 1000 samples and obtain 95% bootstrap percentile CI
  means.0 <- replicate(n = nboot, mean(sample(t0, replace=TRUE)))
  ci.0 <- quantile(means.0, c(alpha/2,1-(alpha/2)))
  means.1 <- replicate(n = nboot, mean(sample(t1, replace=TRUE)))
  ci.1 <- quantile(means.1, c(alpha/2,1-(alpha/2)))

  #Nicely present means and CI bounds for each treatment group
  resultVec <- c(mean(means.0),ci.0, mean(means.1), ci.1)
  names(resultVec) <- c("mean0", "lb0", "ub0", "mean1", "lb1", "ub1")

  return(resultVec)
}
```

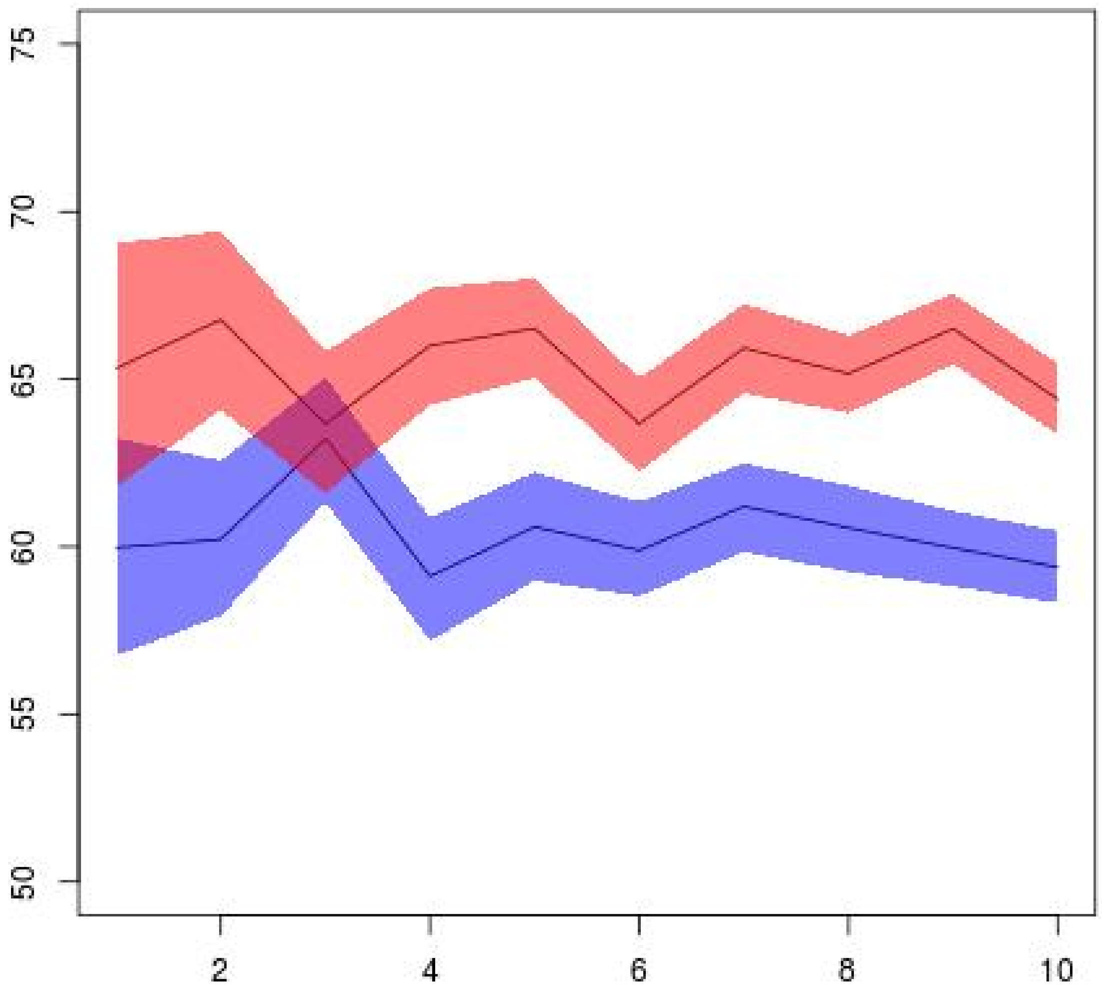
Now, we need to run this function for the varying sample sizes. We store the results from each into a data frame.

```
#Initialize information vectors
sampSize <- numeric(10)
for(i in 1:10){
  sampSize[i] <- 250*i
}

bootMean <- numeric(20)
lowerCI <- numeric(20)
upperCI <- numeric(20)

#Calculate bootstrap mean and CI for varying sample size
for(n in 1:10){
  #Bounds for adding values to info vectors
  a <- (2*n)-1
  b <- 2*n

  #Run function to obtain bootstrapped mean and CI
  x <- bootPercentCI(sampleSize = sampSize[n])

  #Substitute into info vectors
  bootMean[a:b] <- x[c("mean0","mean1")]
  lowerCI[a:b] <- x[c("lb0", "lb1")]
  upperCI[a:b] <- x[c("ub0", "ub1")]
}

#Store all information in data frame
df <- data.frame("Sample.Size" = rep(sampSize, each = 2), "Treatment.Group" = rep(0:1, 10), "Boot.Mean"
df
```

```
##    Sample.Size Treatment.Group Boot.Mean Lower.CI Upper.CI
## 1           250               0  59.94943 56.27479 63.44327
## 2           250               1  61.14541 57.88017 64.35984
## 3           500               0  60.36843 58.00968 62.66002
## 4           500               1  65.59759 62.97057 68.22559
## 5           750               0  60.53442 58.46025 62.68464
## 6           750               1  65.65300 63.66518 67.59453
## 7          1000               0  57.78537 55.84494 59.58304
## 8          1000               1  66.55905 64.83834 68.10613
## 9          1250               0  59.92146 58.25609 61.49055
## 10         1250               1  65.34964 63.91283 66.91087
## 11         1500               0  59.59798 58.18216 61.06845
## 12         1500               1  63.92772 62.46323 65.31396
## 13         1750               0  61.58593 60.35048 62.91765
## 14         1750               1  65.57559 64.17288 66.88938
## 15         2000               0  60.54303 59.29971 61.90517
## 16         2000               1  64.79009 63.55827 66.04475
## 17         2250               0  60.56994 59.37914 61.77023
## 18         2250               1  63.98565 62.80474 65.24158
## 19         2500               0  60.65070 59.51995 61.91564
## 20         2500               1  64.68435 63.65581 65.78065
```

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).



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

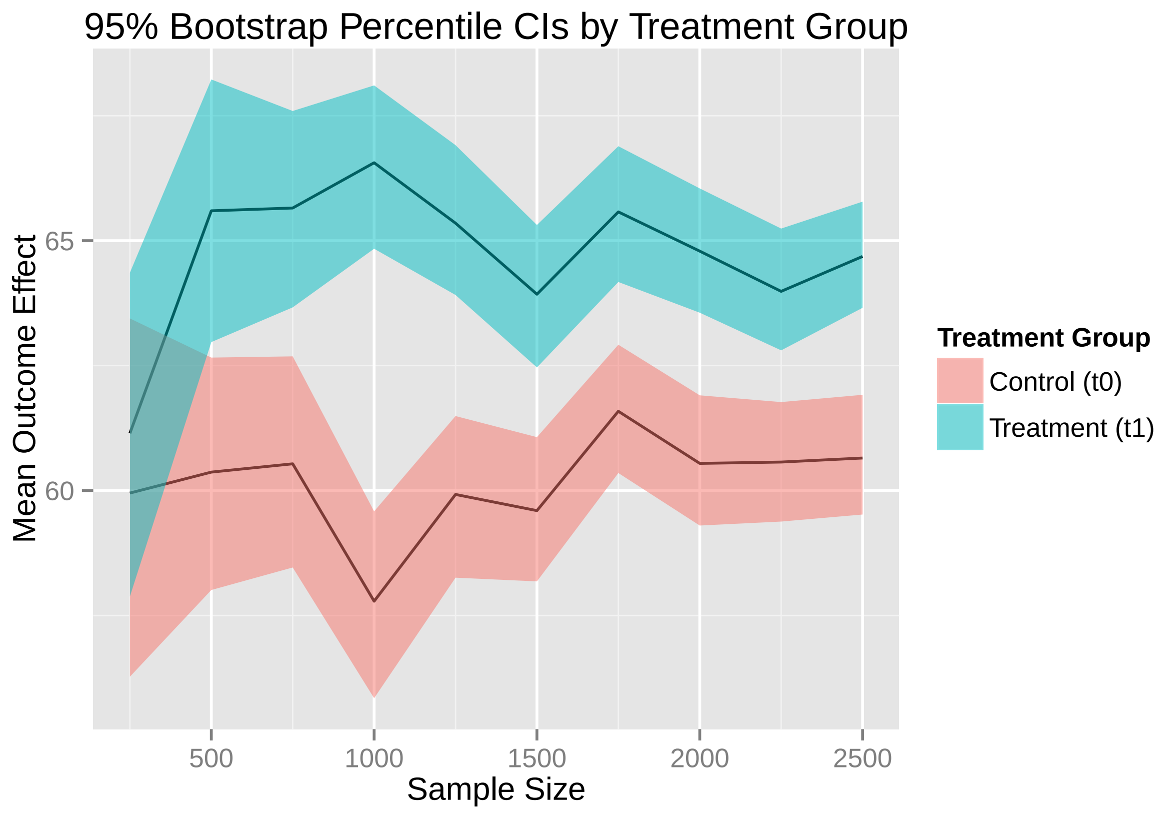
```
makeTransparent = function(..., alpha=0.5) {
  if(alpha<0 | alpha>1) stop("alpha must be between 0 and 1")
  alpha = floor(255*alpha)
  newColor = col2rgb(col=unlist(list(...)), alpha=FALSE)
  .makeTransparent = function(col, alpha) {
    rgb(red=col[1], green=col[2], blue=col[3], alpha=alpha, maxColorValue=255)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

par(new=FALSE)
```

```r
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))
```

Solution

We plot the results using `ggplot2`. Initially, we create a line chart plotting the bootstrapped means by sample size (separated by treatment group). We then add in shaded areas around each treatment group to indicate the confidence intervals.

```r
#Plot bootstrapped means
plot <- ggplot(data = df, aes(x = Sample.Size, y = Boot.Mean, group = Treatment.Group)) + geom_line()
#Add in shaded regions for 95% CIs
plot <- plot + geom_ribbon(data = df, aes(ymin = Lower.CI, ymax = Upper.CI, group = Treatment.Group, fil
#Set title, axis labels, and legend
plot <- plot + labs(title = "95% Bootstrap Percentile CIs by Treatment Group", x = "Sample Size", y = "
#Display plot
plot
```



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 = name, gender = gender, date_of_birth = dob, date_of_admission = doa, pulse = pulse, tempe
}
```

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)

Solution
To make things simpler, instead of redefining the output after running `makePatient()`, I edited the code provided above. The output will be returned with variables (list elements) defined as specified.

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

patient
```

```
## $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"
```

```
class(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)

Solution
We write each of the functions indicated above.

For `mean`:

```
mean.medicalRecord <- function(medRec){
  cat(sprintf("Average Pulse: %s\nAverage Temperature: %s\nAverage Fluid Intake: %s", mean(medRec$pulse
}
```

For `print`:

```
print.medicalRecord <- function(medRec){
  cat(sprintf("Name: %s\nGender: %s\nDate of Birth: %s\n", medRec$name, medRec$gender, medRec$date_of_b
  df <- data.frame(medRec$date_of_admission, medRec$pulse, medRec$temperature, medRec$fluid_intake, row
  names(df) <- c("Date.of.Admission", "Pulse", "Temperature", "Fluid.Intake")
  df <- df[order(df$Date.of.Admission),]
  print(df, row.names = FALSE)
}
```

For `plot`:

```
plot.medicalRecord <- function(medRec){
  df <- data.frame(medRec$date_of_admission, medRec$pulse, medRec$temperature, medRec$fluid_intake)
  names(df) <- c("Date.of.Admission", "Pulse", "Temperature", "Fluid.Intake")
  df <- df[order(df$Date.of.Admission),]
  par(mfrow = c(1,3), oma=c(0,0,1,0))
  plot(df$Date.of.Admission, df$Pulse, type='l', main = "Pulse", xlab = "", ylab = "")
  plot(df$Date.of.Admission, df$Temperature, type='l', main = "Temperature", xlab = "Date of Admission"
  plot(df$Date.of.Admission, df$Fluid.Intake, type='l', main = "Fluid Intake", xlab = "", ylab = "")
  title(main = "Measurements by Date of Admission", outer=TRUE)
}
```

Now we call each of these functions on our `medicalRecord` object.

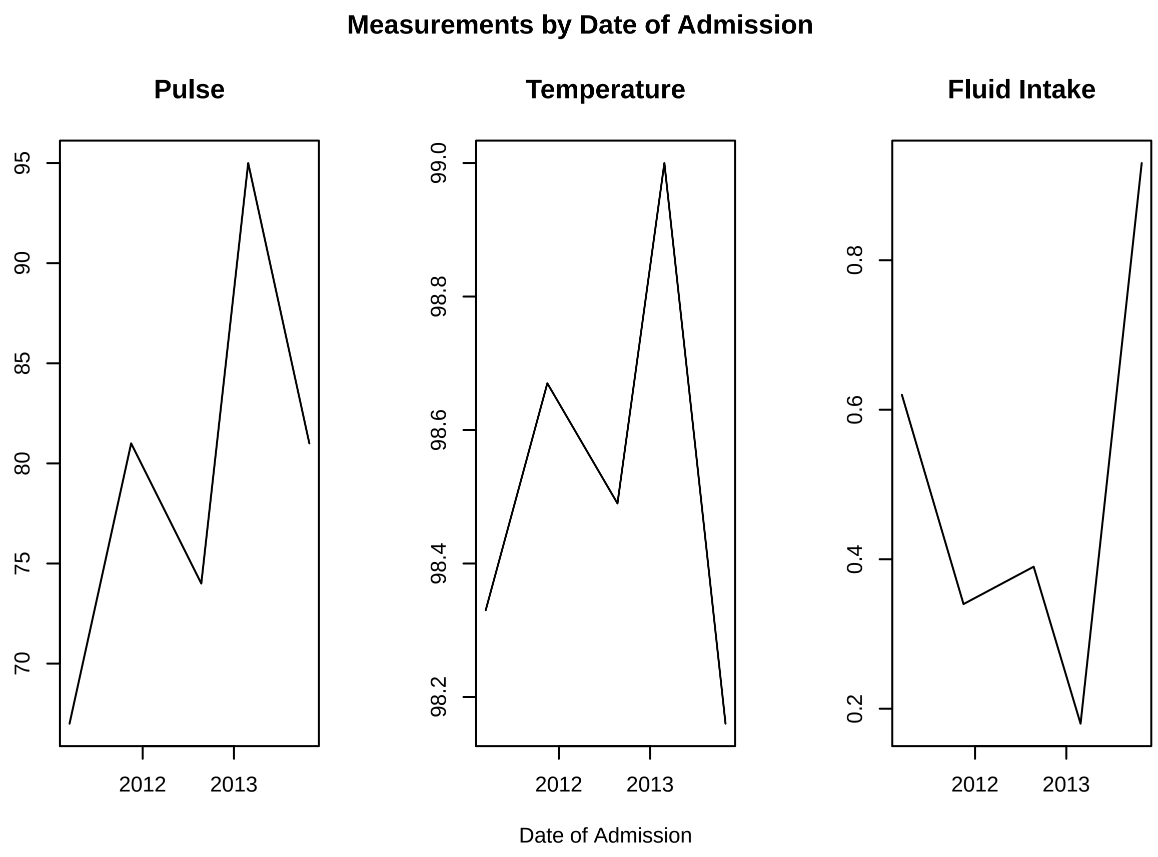
```r
mean(patient)
```

```
## Average Pulse: 79.6
## Average Temperature: 98.53
## Average Fluid Intake: 0.492
```

```r
print(patient)
```

```
## Name: Mev
## Gender: male
## Date of Birth: 1976-08-09
##
##  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
```

```r
plot(patient)
```



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)

Solution
We use the `makePatient` function to create 10 patients, then assign the class of the group to be `cohort`. From there, we adjust the functions `mean` and `print` written in part 2 to be applied to each patient in the cohort. Within each function, we assign the class of each patient to be `medicalRecord`, then call the functions from part 2 on each.

```r
set.seed(8)
group <- replicate(10, list(makePatient()))
class(group) <- 'cohort'

mean.cohort <- function(grp){
  for(patient in grp){
    class(patient) <- 'medicalRecord'
    #Separate each record more cleanly
    cat(sprintf("~Measurements for Patient: %s\n", patient$name))
    cat(mean(patient), "\n")
  }
}

print.cohort <- function(grp){
  for(patient in grp){
    class(patient) <- 'medicalRecord'
    #Separate each record more cleanly
    cat(sprintf("\n~Medical Record"), "\n")
    print(patient)
  }
}
```

Now call each function on our `cohort` object:

```r
mean(group)
```

```
## ~Measurements for Patient: Mev
## Average Pulse: 79.6
## Average Temperature: 98.53
## Average Fluid Intake: 0.492
##
## ~Measurements for Patient: Yul
## Average Pulse: 78
## Average Temperature: 98.495
## Average Fluid Intake: 0.245
##
## ~Measurements for Patient: Zet
## Average Pulse: 81.5
## Average Temperature: 98.44
## Average Fluid Intake: 0.403333333333333
##
## ~Measurements for Patient: Qih
## Average Pulse: 78
## Average Temperature: 98.6
## Average Fluid Intake: 0.65
##
## ~Measurements for Patient: Wut
## Average Pulse: 88.3333333333333
## Average Temperature: 98.05
## Average Fluid Intake: 0.586666666666667
##
## ~Measurements for Patient: Juy
## Average Pulse: 83.5
## Average Temperature: 98.45
## Average Fluid Intake: 0.4525
##
## ~Measurements for Patient: God
## Average Pulse: 83
## Average Temperature: 98.01
## Average Fluid Intake: 0.97
##
## ~Measurements for Patient: Fut
## Average Pulse: 77.5
## Average Temperature: 98.1483333333333
## Average Fluid Intake: 0.336666666666667
##
## ~Measurements for Patient: Pet
## Average Pulse: 77
## Average Temperature: 98.83
## Average Fluid Intake: 0.445
##
## ~Measurements for Patient: Yed
## Average Pulse: 79.3333333333333
## Average Temperature: 98.3
## Average Fluid Intake: 0.658333333333333
##
```

```r
print(group)
```

```
##
## ~Medical Record
## Name: Mev
## Gender: male
## Date of Birth: 1976-08-09
##
##  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
##
## ~Medical Record
## Name: Yul
## Gender: male
## Date of Birth: 1988-06-28
##
##  Date.of.Admission Pulse Temperature Fluid.Intake
##         2012-01-16    76       98.92         0.14
##         2013-08-07    80       98.07         0.35
##
```

```
## ~Medical Record
## Name: Zet
## Gender: female
## Date of Birth: 1970-06-13
##
##  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
##
## ~Medical Record
## Name: Qih
## Gender: female
## Date of Birth: 1987-08-30
##
##  Date.of.Admission Pulse Temperature Fluid.Intake
##         2011-06-22    78        98.6         0.65
##
## ~Medical Record
## Name: Wut
## Gender: male
## Date of Birth: 1974-06-28
##
##  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
##
## ~Medical Record
## Name: Juy
## Gender: male
## Date of Birth: 1983-06-09
##
##  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
##
## ~Medical Record
## Name: God
## Gender: female
## Date of Birth: 1990-02-12
##
##  Date.of.Admission Pulse Temperature Fluid.Intake
##         2010-03-12    83       98.01         0.97
##
## ~Medical Record
## Name: Fut
## Gender: male
## Date of Birth: 1970-01-11
```

```
##
##  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
##
## ~Medical Record
## Name: Pet
## Gender: male
## Date of Birth: 1979-01-01
##
##  Date.of.Admission Pulse Temperature Fluid.Intake
##         2010-10-30    85       98.84         0.60
##         2012-05-10    69       98.82         0.29
##
## ~Medical Record
## Name: Yed
## Gender: male
## Date of Birth: 1977-11-11
##
##  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
```