Bios 301: Assignment 4

Danielle Nemetz

December 6, 2013

1 Question 1

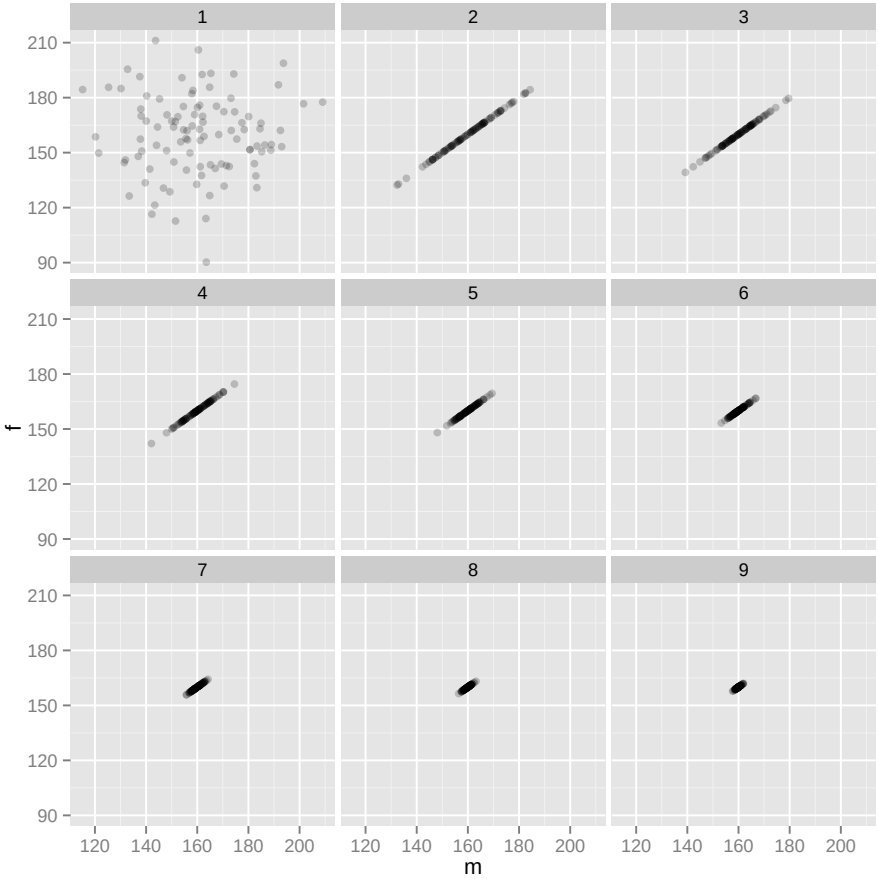
10 points

Use the simulated results from question 3 in assignment 3 to *exactly* reproduce the following plot in ggplot2. Please show your code.:

![generations plot](http://d.pr/i/Xh0d+)

```
library(ggplot2)
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
next_gen <- function(pop) {
    pop$m <- sample(pop$m)
    pop$m <- apply(pop, 1, mean)
    pop$f <- pop$m
    return(pop)
}
# 900 rows, column for height and column for generation
D <- data.frame(generation = rep(1:9, each = 100), m = c(0), f = c(0))
# Creating generations 1-9
pop1 <- next_gen(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)
D[1:100, 2] <- pop[, 1]
D[101:200, 2] <- pop2[, 1]
D[201:300, 2] <- pop3[, 1]
D[301:400, 2] <- pop4[, 1]
D[401:500, 2] <- pop5[, 1]
D[501:600, 2] <- pop6[, 1]
D[601:700, 2] <- pop7[, 1]
```

```
D[701:800, 2] <- pop8[, 1]
D[801:900, 2] <- pop9[, 1]
D[1:100, 3] <- pop[, 2]
D[101:200, 3] <- pop2[, 2]
D[201:300, 3] <- pop3[, 2]
D[301:400, 3] <- pop4[, 2]
D[401:500, 3] <- pop5[, 2]
D[501:600, 3] <- pop6[, 2]
D[601:700, 3] <- pop7[, 2]
D[701:800, 3] <- pop8[, 2]
D[801:900, 3] <- pop9[, 2]
# not getting the correct plot
qplot(m, f, data = D, facets = ~generation, alpha = I(1/5))
```



```
# change intensity to fix color, and fix labels, and axis increments
```

2 Question 2

6 points

Approximate the probability that the proportion of heads obtained will be between 0.50 and 0.52 when a fair coin is tossed

1. 50 times. 2. 500 times.

```
# H0: P=0.5, H1: P/=0.5 Want to know p(0.50 < p < 0.52) Compute P(.50 < X/50
# < .52) = P(25 < x < 26) Standardize: (25-25)/sqrt(50*.5*.5)<z<
# (26-25)/sqrt(50*.5*.5)
pnorm(0.2828) - pnorm(0)

## [1] 0.1113

# 2. 500 times: Compute P(.50 < X/500 < .52) = P(250 < x < 260) Standardize:
# (250-250)/sqrt(500*.5*.5)<z< (260-250)/sqrt(500*.5*.5)
pnorm(0.894) - pnorm(0)

## [1] 0.3143
```

3 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

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

```
n <- 100
u <- runif(n, min = -1, max = 1)
mean(u)

## [1] 0.1417

upper.bound <- mean(u) + 1.96 * sd(u)/sqrt(n)
lower.bound <- mean(u) - 1.96 * sd(u)/sqrt(n)
mean <- NULL
```

```r
upper.bound <- NULL
lower.bound <- NULL
contains.mean <- NULL
for (i in 1:1000) {
    n <- 100
    u <- runif(n, min = -1, max = 1)
    mean[i] <- mean(u)
    upper.bound[i] <- mean[i] + 1.96 * sd(u)/sqrt(n)
    lower.bound[i] <- mean[i] - 1.96 * sd(u)/sqrt(n)
    if (lower.bound[i] < 0 && upper.bound[i] > 0)
        contains.mean[i] <- 1 else contains.mean[i] <- 0
}
Data <- data.frame(mean, lower.bound, upper.bound, contains.mean)
colnames(Data) <- c("Sample Mean", "Lower Bound", "Upper Bound", "Contains Mean")
mean_contains.mean <- mean(contains.mean)
cat("Number of trials: 1000")

## Number of trials: 1000

head(Data)

##   Sample Mean Lower Bound Upper Bound Contains Mean
## 1  -0.0318574    -0.14349     0.07978             1
## 2  -0.0214710    -0.13162     0.08868             1
## 3   0.0163870    -0.09140     0.12417             1
## 4   0.0700756    -0.04797     0.18812             1
## 5  -0.0744693    -0.19410     0.04517             1
## 6  -0.0009474    -0.11783     0.11593             1

cat(mean(contains.mean) * 100, "percent of CI's contained the mean")

## 94.3 percent of CI's contained the mean
```

4 Question 4

24 points

Programming with classes:

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 (Hint: you may need to use a vector or data frame somewhere).

```
Jeffery <- list(name = "Jeffery", gender = "Male", date_of_birth = "03/14/1970",
    date_of_admission = c("05/30/2012", "07/06/2011"), pulse = c(99, 95), temperature = c(98
        99.1), fluid_intake = c(16, 16.5))
Robert <- list(name = "Robert", gender = "Male", date_of_birth = "08/28/1961",
    date_of_admission = c("10/28/2012", "04/03/2013"), pulse = c(104, 101),
    temperature = c(103.2, 100.1), fluid_intake = c(4, 5.5))
Chris <- list(name = "Chris", gender = "Male", date_of_birth = as.Date("01/25/1975",
    "%m/%d/%Y"), date_of_admission = "06/12/2013", pulse = c(70, 68), temperature = 98.5,
    fluid_intake = 16)
class(Jeffery) <- "medicalRecord"
class(Chris) <- "medicalRecord"
class(Robert) <- "medicalRecord"
```

2. Write a ‘medicalRecord‘ method for the generic function ‘mean‘, which returns averages for pulse, temperature and fluids. Also write a ‘medicalRecord‘ method either for ‘print‘, which employs some nice formatting, perhaps arranging measurements by date, or ‘plot‘ that generates a composite plot of measurements over time.

```
mean.medicalRecord <- function(patient) {
    mean.pulse <- mean(patient$pulse)
    mean.temperature <- mean(patient$temperature)
    mean.fluid_intake <- mean(patient$fluid_intake)
    return(c(mean.pulse, mean.temperature, mean.fluid_intake))
}
# Verify function works:
mean.medicalRecord(Jeffery)

## [1] 97.00 98.90 16.25

print.medicalRecord <- function(patient) {
    cat("name:", patient$name, "\n")
    cat("gender:", patient$gender, "\n")
    cat("date of birth:", patient$date_of_birth, "\n")
    cat("date of admission:", patient$date_of_admission, "\n")
    cat("pulse:", patient$pulse, "\n")
    cat("temperature:", patient$temperature, "\n")
    cat("fluid intake:", patient$fluid_intake, "\n")
    cat("average pulse:", mean.medicalRecord(patient)[1], "\n")
    cat("average temperature:", mean.medicalRecord(patient)[2], "\n")
    cat("average fluid intake:", mean.medicalRecord(patient)[3], "\n")
}
# Verify print works:
Jeffery

## name: Jeffery
```

```
## gender: Male
## date of birth: 03/14/1970
## date of admission: 05/30/2012 07/06/2011
## pulse: 99 95
## temperature: 98.7 99.1
## fluid intake: 16 16.5
## average pulse: 97
## average temperature: 98.9
## average fluid intake: 16.25
```

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.

```
Cohort <- list(Jeffery, Chris, Robert)
class(Cohort) <- "Group"
mean.Group <- function(patients) {
    mean.pulse <- NULL
    mean.temperature <- NULL
    mean.fluid_intake <- NULL
    for (i in 1:length(patients)) {
        mean.pulse[i] <- mean.medicalRecord(patients[[i]])[1]
        mean.temperature[i] <- mean.medicalRecord(patients[[i]])[2]
        mean.fluid_intake[i] <- mean.medicalRecord(patients[[i]])[3]
    }
    mean2.pulse <- mean(mean.pulse)
    mean2.temperature <- mean(mean.temperature)
    mean2.fluid_intake <- mean(mean.fluid_intake)
    return(c(mean2.pulse, mean2.temperature, mean2.fluid_intake))
}
print.Group <- function(patients) {
    cat("The number of people in the cohort is:", length(patients), "\n")
    cat("Mean pulse for the cohort is:", mean.Group(patients)[1], "\n")
    cat("Mean temperature for the cohort is:", mean.Group(patients)[2], "\n")
    cat("Mean fluid intake for the cohort is:", mean.Group(patients)[3], "\n")
}
# Verify this works:
Cohort

## The number of people in the cohort is: 3
## Mean pulse for the cohort is: 89.5
## Mean temperature for the cohort is: 99.68
## Mean fluid intake for the cohort is: 12.33
```