Stat Comp hw 6

Michael Ackerman

December 3, 2015

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:

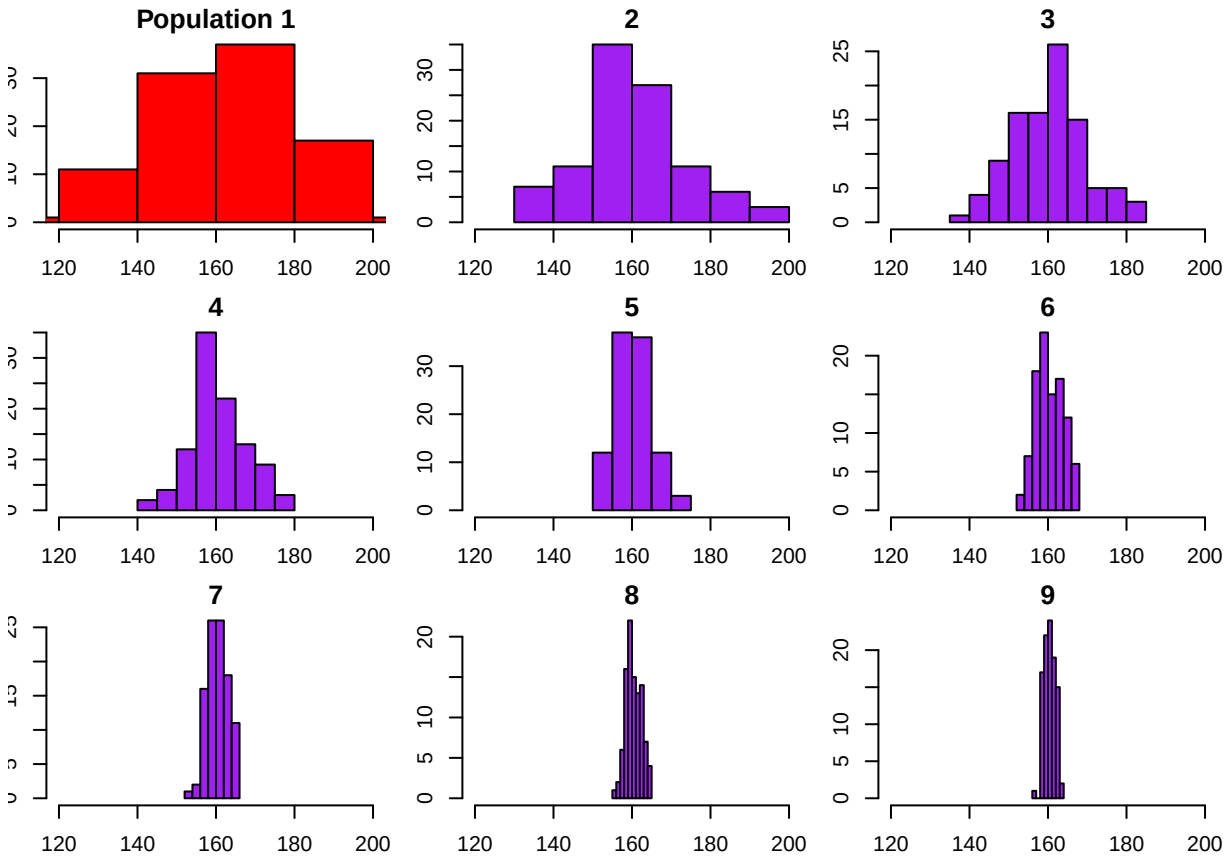
```
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
```

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.

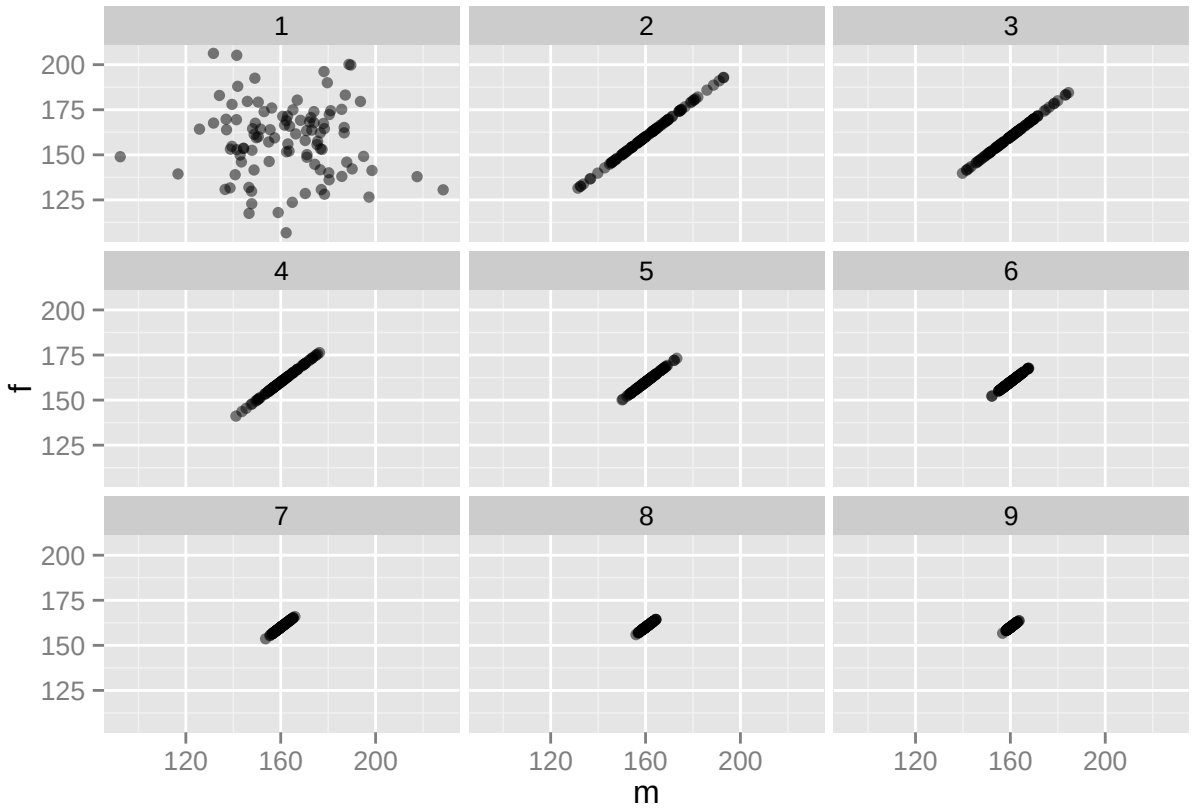
```
pop <- data.frame(m = rnorm(100, 160, 20), f = rnorm(100, 160, 20))
gen<- as.data.frame(pop)
par(mfrow=c(3,3), mai=c(0.3,0.2,0.2,0.2))
hist(pop$m, main= "Population 1", xlab=NULL, xlim=c(120,200), col="red")
for(i in 2:9){
pop <- next_gen(pop)
hist(pop$m, main= i, xlab=NULL, xlim=c(120,200), col="purple" )
gen <- rbind(gen,pop)
}
```



Question 2

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

```
gen$children <- rep(1:9,each=100)
library(ggplot2)
ggplot(gen) + geom_point(alpha = .5) + aes(x=m, y=f) + facet_wrap(~children, 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)

```
ss<-seq(250,2500,250)
boot0 <- boot1 <-LB0 <- LB1 <- UB0 <- UB1 <- numeric(length(ss))

for(i in seq_along(ss)){
  pop <- sample(0:1,ss[i], replace=T)
  g0 <- sample(rnorm(length(which(pop==0)), 60, 20), ss[i], replace = T)
  g1 <- sample(rnorm(length(which(pop==1)), 65, 20), ss[i], replace = T)

  boot0[i] <- mean(g0)
  boot1[i] <- mean(g1)

c0 <- c1 <-0
for(j in 1:1000){
  c0[j] <- mean(sample(g0,replace=T))
```

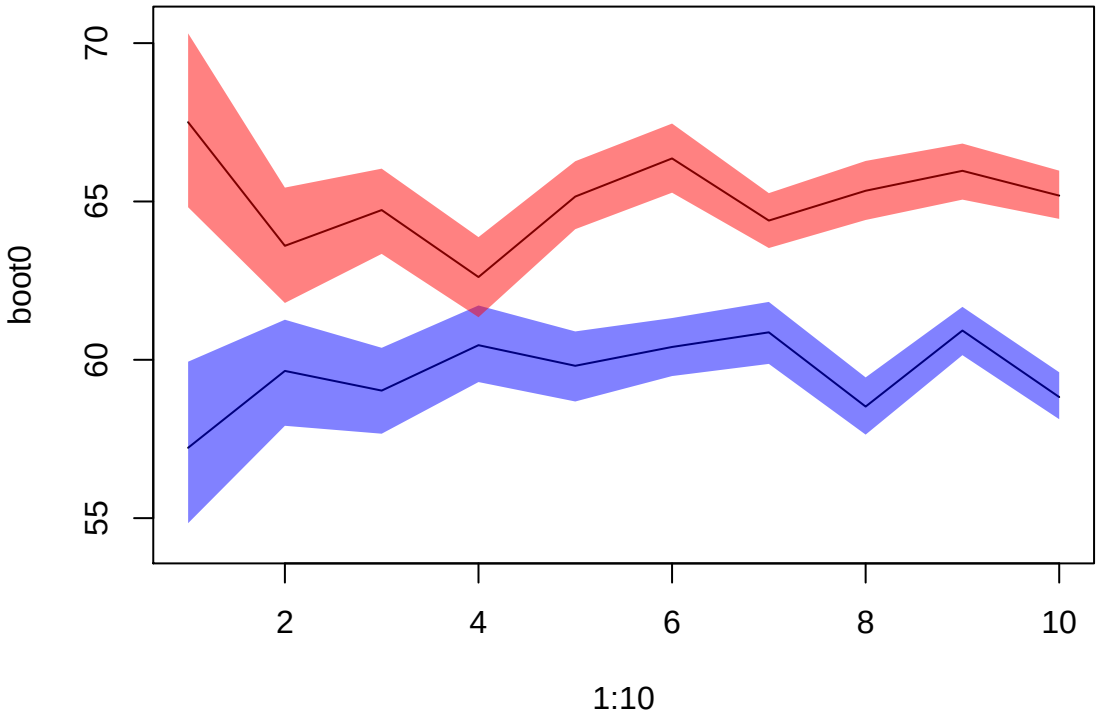
```
  c1[j] <- mean(sample(g1,replace=T))
}

  LB0[i] <- quantile(c0,.025)
  UB0[i] <- quantile(c0,.975)
  LB1[i] <- quantile(c1,.025)
  UB1[i] <- quantile(c1,.975)
}


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(mfrow=c(1,1))
par(mai=c(0.8,0.8,0.8,0.8))
plot(1:10,boot0,type="l", ylim= c(min(boot0)-3, max(boot1)+3))
lines(1:10,boot1)

rbind(LB0,boot0, UB0)
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## LB0   54.83980 57.91602 57.66602 59.29742 58.68324 59.48618 59.87071
## boot0 57.22195 59.64921 59.02772 60.46158 59.81043 60.40558 60.86733
## UB0   59.94051 61.26610 60.37912 61.71541 60.89639 61.31884 61.82808
##           [,8]     [,9]    [,10]
## LB0   57.63757 60.14568 58.12284
## boot0 58.52286 60.92412 58.82417
## UB0   59.44442 61.67155 59.60647
```

```
rbind(LB1,boot1, UB1)
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## LB1   64.81801 61.79510 63.34391 61.34468 64.12501 65.27678 63.52972
## boot1 67.50265 63.59967 64.72827 62.61337 65.15481 66.35854 64.39690
## UB1   70.30918 65.43738 66.03647 63.87502 66.27138 67.46076 65.26315
##           [,8]     [,9]    [,10]
## LB1   64.41556 65.05788 64.44968
## boot1 65.33815 65.96789 65.18511
## UB1   66.27968 66.82911 65.97291
```

```
polygon(x=c(1:10,10:1), y= c(LB0,rev(UB0)) , border=NA, col=makeTransparent('blue',alpha=0.5))
polygon(x=c(1:10,10:1), y= c(LB1,rev(UB1)) , border=NA, col=makeTransparent('red',alpha=0.5))
```



Question 4

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, dob=dob, doa=doa, pulse=pulse, temp=temp, fluid=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)
per <- makePatient()
class(per) <- '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)

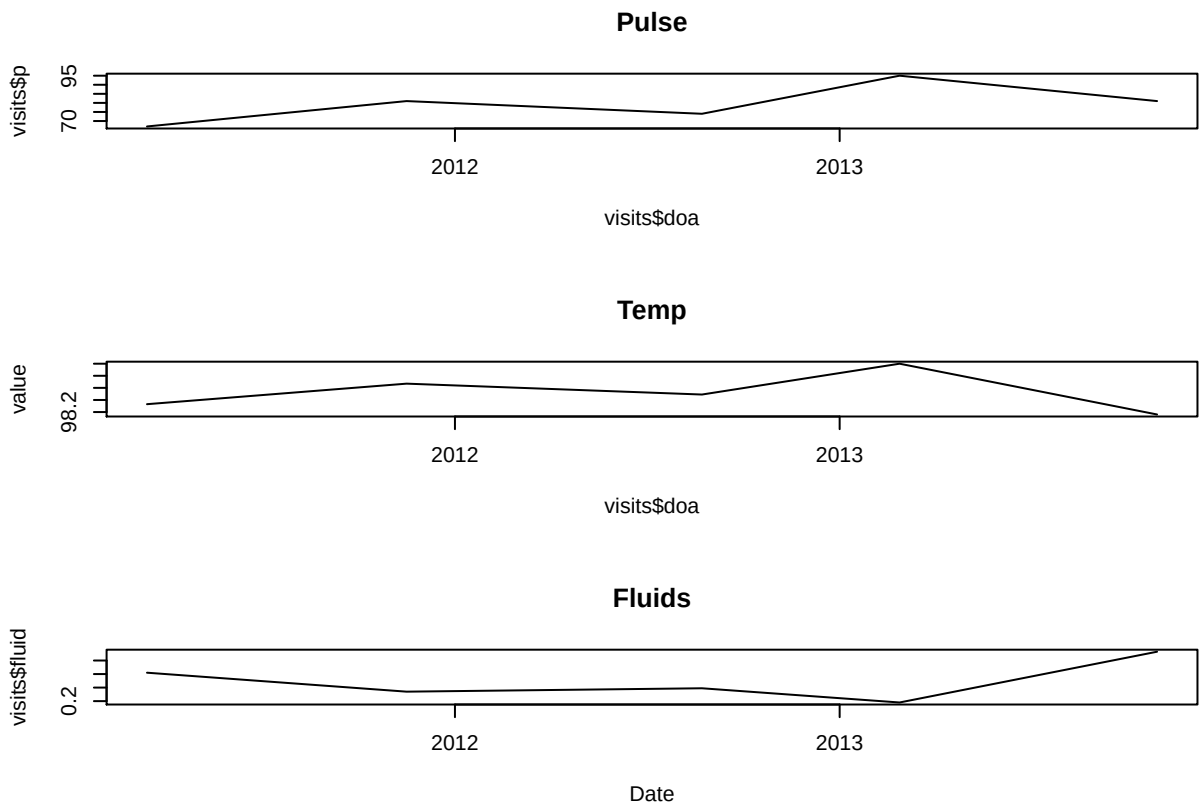
```
mean.medicalRecord <- function(x){
  mean.p <- mean(x$pulse)
  mean.t <- mean(x$temp)
  mean.f <- mean(x$fluid)

  return(list(mean.p = mean.p,mean.t = mean.t,mean.f = mean.f))
  }

print.medicalRecord <- function(x){
  cat(paste("Name:", x$name, "\n"))
  cat(sprintf("%s patient born on %s", x$gender, x$dob),"\n")
  cat("Visits:\n")

visits <- data.frame(doa = x$doa, p = x$pulse, t = x$temp, fluid = x$fluid)
  visits = visits[order(visits$doa), ]
  for(i in 1:length(visits$doa)){
    cat(sprintf("Date: %s | Pulse: %3.2s | Temp: %6.4s | Fluid Intake: %6.4s\n",
      visits$doa[i], visits$p[i], visits$t[i], visits$fluid[i]))
  }
}

plot.medicalRecord <- function(x){
  visits = data.frame(doa = x$doa, p = x$pulse, t = x$temp, fluid = x$fluid)
  visits = visits[order(visits$doa), ]
  par(mfrow = c(3,1))
  plot(visits$doa, visits$p, type = "l", xlab = NULL, ylab = NULL, main = "Pulse")
  plot(visits$doa, visits$t, type = "l", xlab = NULL, ylab = "value", main = "Temp")
  plot(visits$doa, visits$fluid, type = "l", xlab = "Date", ylab = NULL, main = "Fluids")
}

mean(per)
```

```
## $mean.p
## [1] 79.6
##
## $mean.t
## [1] 98.53
##
## $mean.f
## [1] 0.492
```

```
print(per)
```

```
## Name: Mev
## male patient born on 1976-08-09
## 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
```

```
plot(per)
```



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)

c <- list()
for(i in 1:10){
  c[[i]] <- patient <- makePatient()
  }
class(c) <- "cohort"

mean.cohort = function(c){
  for(patient in c){
    class(patient) = "medicalRecord"
    cat("------------------------\n")
    print(mean(patient))
  }
}

print.cohort = function(c){
  for(patient in c){
    class(patient) = "medicalRecord"
    cat("------------------------\n")
```

```
    print(patient)
  }
}

mean(c)
```

```
## ----------------------
## $mean.p
## [1] 79.6
##
## $mean.t
## [1] 98.53
##
## $mean.f
## [1] 0.492
##
## ----------------------
## $mean.p
## [1] 78
##
## $mean.t
## [1] 98.495
##
## $mean.f
## [1] 0.245
##
## ----------------------
## $mean.p
## [1] 81.5
##
## $mean.t
## [1] 98.44
##
## $mean.f
## [1] 0.4033333
##
## ----------------------
## $mean.p
## [1] 78
##
## $mean.t
## [1] 98.6
##
## $mean.f
## [1] 0.65
##
## ----------------------
## $mean.p
## [1] 88.33333
##
## $mean.t
## [1] 98.05
##
```

```
## $mean.f
## [1] 0.5866667
##
## ----------------------
## $mean.p
## [1] 83.5
##
## $mean.t
## [1] 98.45
##
## $mean.f
## [1] 0.4525
##
## ----------------------
## $mean.p
## [1] 83
##
## $mean.t
## [1] 98.01
##
## $mean.f
## [1] 0.97
##
## ----------------------
## $mean.p
## [1] 77.5
##
## $mean.t
## [1] 98.14833
##
## $mean.f
## [1] 0.3366667
##
## ----------------------
## $mean.p
## [1] 77
##
## $mean.t
## [1] 98.83
##
## $mean.f
## [1] 0.445
##
## ----------------------
## $mean.p
## [1] 79.33333
##
## $mean.t
## [1] 98.3
##
## $mean.f
## [1] 0.6583333
```

```r
print(c)
```

```
## ----------------------
## Name: Mev
## male patient born on 1976-08-09
## 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
## ----------------------
## Name: Yul
## male patient born on 1988-06-28
## Visits:
## Date: 2012-01-16 | Pulse:  76 | Temp:   98.9 | Fluid Intake:   0.14
## Date: 2013-08-07 | Pulse:  80 | Temp:   98.0 | Fluid Intake:   0.35
## ----------------------
## Name: Zet
## female patient born on 1970-06-13
## Visits:
## Date: 2010-03-21 | Pulse:  79 | Temp:   98.5 | Fluid Intake:   0.22
## Date: 2010-04-01 | Pulse:  73 | Temp:   98.3 | Fluid Intake:   0.61
## Date: 2012-08-29 | Pulse:  88 | Temp:   98.4 | Fluid Intake:   0.59
## Date: 2013-06-01 | Pulse:  84 | Temp:   98.2 | Fluid Intake:   0.25
## Date: 2013-11-03 | Pulse:  72 | Temp:   98.5 | Fluid Intake:   0.03
## Date: 2014-02-05 | Pulse:  93 | Temp:   98.5 | Fluid Intake:   0.72
## ----------------------
## Name: Qih
## female patient born on 1987-08-30
## Visits:
## Date: 2011-06-22 | Pulse:  78 | Temp:   98.6 | Fluid Intake:   0.65
## ----------------------
## Name: Wut
## male patient born on 1974-06-28
## Visits:
## Date: 2010-04-12 | Pulse:  76 | Temp:   98.0 | Fluid Intake:   0.65
## Date: 2011-02-16 | Pulse:  93 | Temp:   98.2 | Fluid Intake:   0.97
## Date: 2012-04-12 | Pulse:  96 | Temp:   97.8 | Fluid Intake:   0.14
## ----------------------
## Name: Juy
## male patient born on 1983-06-09
## Visits:
## Date: 2010-03-10 | Pulse:  81 | Temp:   99.1 | Fluid Intake:   0.66
## Date: 2010-03-25 | Pulse:  90 | Temp:   98.5 | Fluid Intake:   0.26
## Date: 2010-04-18 | Pulse:  75 | Temp:   98.5 | Fluid Intake:    0.6
## Date: 2010-06-10 | Pulse:  88 | Temp:   97.5 | Fluid Intake:   0.29
## ----------------------
## Name: God
## female patient born on 1990-02-12
## Visits:
## Date: 2010-03-12 | Pulse:  83 | Temp:   98.0 | Fluid Intake:   0.97
## ----------------------
```

```
## Name: Fut
## male patient born on 1970-01-11
## Visits:
## Date: 2011-04-07 | Pulse:  80 | Temp:   97.8 | Fluid Intake:   0.36
## Date: 2011-04-14 | Pulse:  83 | Temp:   97.9 | Fluid Intake:      0
## Date: 2011-08-16 | Pulse:  66 | Temp:   98.4 | Fluid Intake:   0.13
## Date: 2013-03-15 | Pulse:  74 | Temp:   98.3 | Fluid Intake:   0.31
## Date: 2013-06-20 | Pulse:  74 | Temp:   98.4 | Fluid Intake:   0.49
## Date: 2013-11-12 | Pulse:  88 | Temp:   97.8 | Fluid Intake:   0.73
## ----------------------
## Name: Pet
## male patient born on 1979-01-01
## Visits:
## Date: 2010-10-30 | Pulse:  85 | Temp:   98.8 | Fluid Intake:    0.6
## Date: 2012-05-10 | Pulse:  69 | Temp:   98.8 | Fluid Intake:   0.29
## ----------------------
## Name: Yed
## male patient born on 1977-11-11
## Visits:
## Date: 2010-01-28 | Pulse:  63 | Temp:   97.9 | Fluid Intake:   0.94
## Date: 2010-03-06 | Pulse:  81 | Temp:   98.4 | Fluid Intake:   0.67
## Date: 2010-07-10 | Pulse:  98 | Temp:   98.6 | Fluid Intake:   0.79
## Date: 2010-08-27 | Pulse:  66 | Temp:   97.6 | Fluid Intake:   0.36
## Date: 2011-06-18 | Pulse:  83 | Temp:     98 | Fluid Intake:   0.69
## Date: 2013-01-06 | Pulse:  85 | Temp:   99.0 | Fluid Intake:    0.5
```