Bios 6301: Assignment 8

Lydia Yao

Due Tuesday, 16 November, 1:00 PM

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

30 points total.

Submit a single knitr file (named `homework8.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 `homework8.rmd` or include author name may result in 5 points taken off.

Question 1

15 points

Install the `readxl` package and run the following

```
library(readxl)
fn <- 'icd10.xlsx'
if(file.access(fn, mode = 4) == -1) {
    url <- "https://www.cdc.gov/nhsn/xls/icd10-pcs-pcm-nhsn-opc.xlsx"
    download.file(url, destfile = fn, mode = 'wb')
}
dat <- readxl::read_excel(fn, sheet = 2)
```

1. Show the class of `dat`. (1 point)

```
class(dat)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

2. Show the methods available for objects of the given class (if there are multiple classes, show methods for all classes). (3 points)

```
methods(class = 'data.frame')
```

```
##  [1] [             [[            [[<-          [<-           $<-
##  [6] aggregate     anyDuplicated anyNA         as.data.frame as.list
## [11] as.matrix     by            cbind         coerce        dim
## [16] dimnames      dimnames<-    droplevels    duplicated    edit
## [21] format        formula       head          initialize    is.na
## [26] Math          merge         na.exclude    na.omit       Ops
## [31] plot          print         prompt        rbind         row.names
## [36] row.names<-   rowsum        show          slotsFromS3   split
## [41] split<-       stack         str           subset        summary
## [46] Summary       t             tail          transform     type.convert
## [51] unique        unstack       within        xtfrm
## see '?methods' for accessing help and source code
```

```
methods(class = 'tbl')
```

```
##  [1] [[<-        [<-         $<-         coerce      format      initialize
##  [7] Ops         print       show        slotsFromS3
## see '?methods' for accessing help and source code
```

```
methods(class = 'tbl_df')
```

```
##  [1] [             [[            [[<-          [<-           $
##  [6] $<-           as.data.frame coerce        initialize    names<-
## [11] Ops           row.names<-   show          slotsFromS3   str
## see '?methods' for accessing help and source code
```

3. If you call `print(dat)`, what print method is being dispatched? (1 point)

`The print method will dispatch the print under tbl_df.`

4. Set the class of `dat` to be a data.frame. (1 point)

```
class(dat) <- c('data.frame')
```

5. If you call `print(dat)` again, what print method is being dispatched? (1 point)

`This time, the print method will dispatch the print under data.frame.`

Define a new generic function `nUnique` with the code below.

```
nUnique <- function(x) {
    UseMethod('nUnique')
}
```

6. Write a default method for `nUnique` to count the number of unique values in an element. (2 points)

```
nUnique <- function(x) {
  length(unique(x))
}
```

7. Check your function (2 points)

```
nUnique(letters) # should return 26
nUnique(sample(10, 100, replace = TRUE)) # should return 10 (probably)
```

8. Write a data.frame method for `nUnique` to operate on data.frame objects. This version should return counts for each column in a data.frame. (2 points)

```
nUnique.data.frame <- function(x) {
  output <- matrix(NA, nrow = 1, ncol = ncol(x))
  for(i in seq(ncol(x))){
    output[,i] <- length(unique(x[,i]))
  }
  colnames(output) <- names(x)
  output
}
```

9. Check your function (2 points)

```
nUnique.data.frame(dat)
```

Question 2

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, 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)
j <- makePatient()
class(j) <- 'medicalRecord'
print(j)
```

```
## [[1]]
## [1] "Yes"
##
## [[2]]
## [1] male
## Levels: female male
##
## [[3]]
## [1] "1977-05-03"
##
## [[4]]
## [1] "2013-06-09" "2013-07-02"
##
## [[5]]
## [1] 79 78
##
## [[6]]
## [1] 98.07 97.50
##
## [[7]]
## [1] 0.28 0.52
##
## 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)

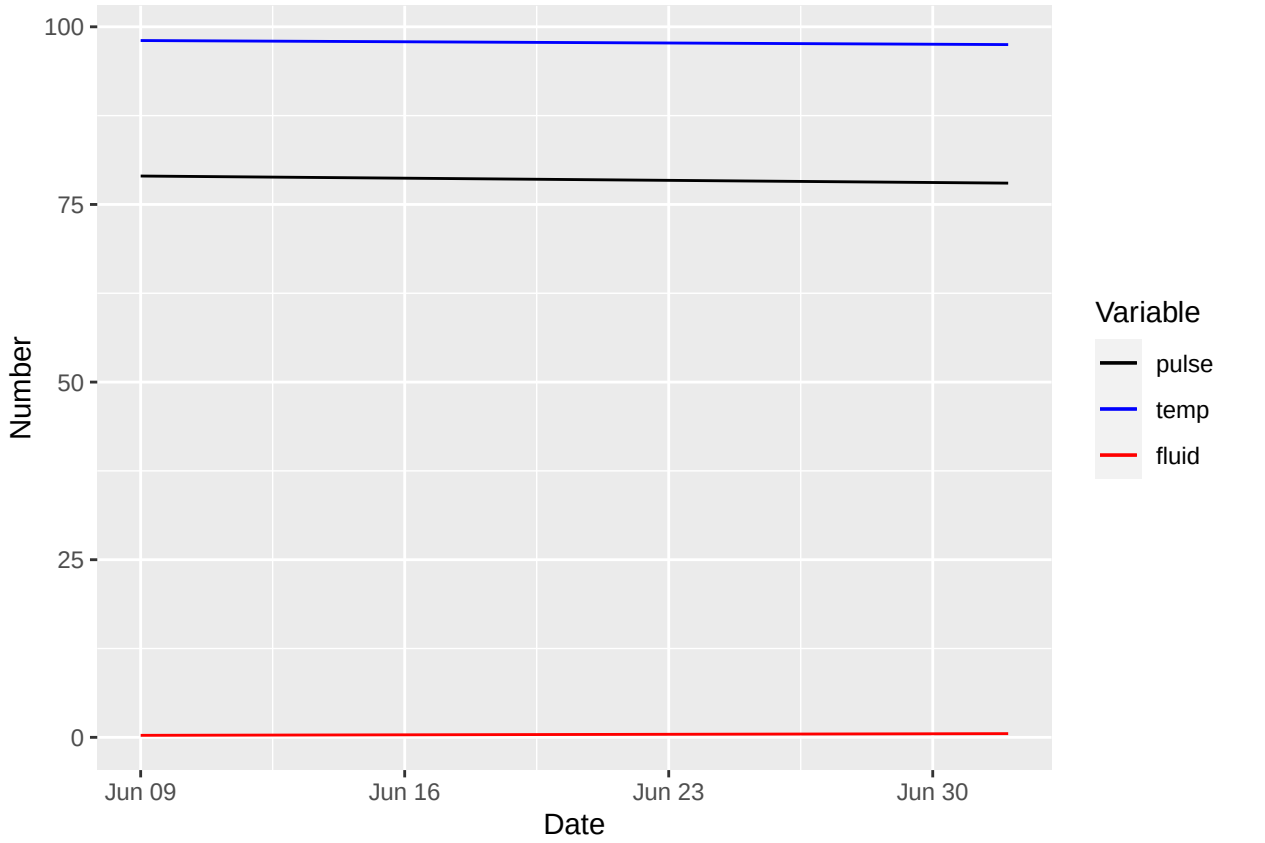
```r
mean.medicalRecord <- function(x){
  output <- matrix(NA, nrow = 1, ncol = 3)
  x <- x[c(5,6,7)]
  for(i in 1:3){
    output[,i] <- sapply(x[i], mean, na.rm = TRUE)
  }
  colnames(output) <- c("pulse", "temperature", "fluids")
  output
}

print.medicalRecord <- function(x){
  if(x[2] == 2) {gender = "male"}
  else{gender = "female"}

  dob = as.Date(as.integer(as.character.POSIXt(x[3])), origin="2010-01-01")

  four = sapply(x[4], as.character.POSIXt)
  four = sort(four)
  four = paste(four, collapse = ", ")

  cat(sprintf("name: %s\ngender: %s\ndob: %s\ndoa: %s\npulse: %s\ntemp: %s\nfluid: %s",
              x[1], gender, dob, four, substr(x[5],3,nchar(x[5]) - 1), substr(x[6],3,nchar(x[6]) - 1), s
}
library(ggplot2)
plot.medicalRecord <- function(x){

  dates = sapply(x[4], as.character.POSIXt)
  dates = sapply(dates, as.Date)
  data <- data.frame(
    doa = as.Date(dates, origin="2010-01-01"),
    pulse = x[5],
    temp = x[6],
    fluid = x[7]
  )
  colnames(data) <- c("doa", "pulse", "temp", "fluid")
  ggplot(data) + geom_line(aes(x = doa, y = pulse, color = "black")) + geom_line(aes(x = doa, y = temp,
                           breaks = c("black", "blue", "red"),
                           labels = c("pulse", "temp", "fluid"),
                           guide = "legend") + xlab("Date") + ylab("Number")
}
mean(j)
```

```
##      pulse temperature fluids
## [1,]  78.5      97.785    0.4
```

```r
print(j)
```

```
## name: Yes
## gender: male
## dob: 2017-05-03
## doa: 2013-06-09, 2013-07-02
## pulse: 79, 78
## temp: 98.07, 97.5
## fluid: 0.28, 0.52
```

```
plot(j)
```



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)
cohort <- function(n){
  result = makePatient()
  for(i in 1:(n-1)){
    result = c(result, makePatient())
  }
  result
}
j2 <- cohort(10)
class(j2) <- 'cohortRecord'

mean.cohortRecord <- function(x){
  output <- matrix(NA, nrow = length(x)/7, ncol = 3)
  pulse = x[seq(5, length(x), 7)]
  temp = x[seq(6, length(x), 7)]
  fluids = x[seq(7, length(x), 7)]
  for(i in 1:as.integer(length(x)/7)){
    output[i,1] <- sapply(pulse[i], mean, na.rm = TRUE)
    output[i,2] <- sapply(temp[i], mean, na.rm = TRUE)
    output[i,3] <- sapply(fluids[i], mean, na.rm = TRUE)
```

```
  }
  colnames(output) <- c("pulse", "temperature", "fluids")
  output
}

print.cohortRecord <- function(x){
  count = 1
  for(i in 1:length(x)){
    if(count ==1){
      print(paste("name: ", x[i]))
      count = count + 1
    } else if (count ==2) {
      if(x[i] == 2) {print("gender: male ")}
      else{print("gender: female ")}
      count = count + 1
    } else if (count ==3){
      dob = as.Date(as.integer(as.character.POSIXt(x[i])), origin="2010-01-01")
      print(paste("dob: ", dob))
      count = count + 1
    } else if (count ==4){
      four = sapply(x[i], as.character.POSIXt)
      four = sort(four)
      four = paste(four, collapse = ", ")
      print(paste("doa: ", four))
      count = count + 1
    } else if (count ==5){
      print(paste("pulse: ", substr(x[i],3,nchar(x[i]) - 1)))
      count = count + 1
    } else if (count ==6){
      print(paste("temp: ", substr(x[i],3,nchar(x[i]) - 1)))
      count = count + 1
    } else if (count ==7){
      print(paste("fluid: ", substr(x[i],3,nchar(x[i]) - 1)))
      count = 1
      cat("\n")
    }
  }
}
mean.cohortRecord(j2)
```

```
##          pulse temperature    fluids
##  [1,] 78.50000    97.78500 0.4000000
##  [2,] 86.33333    98.39667 0.4133333
##  [3,] 77.00000    98.64750 0.5200000
##  [4,] 83.16667    98.48500 0.2966667
##  [5,] 83.50000    98.45000 0.4525000
##  [6,] 84.40000    98.48400 0.5220000
##  [7,] 76.50000    98.38000 0.3975000
##  [8,] 75.00000    98.36750 0.5225000
##  [9,] 73.00000    98.36000 0.1500000
## [10,] 77.00000    98.54000 0.1500000
```

```
print.cohortRecord(j2)
```

```
## [1] "name:  Yes"
## [1] "gender: male "
## [1] "dob:  2017-05-03"
## [1] "doa:  2013-06-09, 2013-07-02"
## [1] "pulse:  79, 78"
## [1] "temp:  98.07, 97.5"
## [1] "fluid:  0.28, 0.52"
##
## [1] "name:  Fal"
## [1] "gender: male "
## [1] "dob:  2028-05-24"
## [1] "doa:  2010-11-16, 2013-03-24, 2013-09-12"
## [1] "pulse:  76, 96, 87"
## [1] "temp:  98.23, 98.75, 98.21"
## [1] "fluid:  0.18, 0.96, 0.1"
##
## [1] "name:  Zog"
## [1] "gender: male "
## [1] "dob:  2028-12-14"
## [1] "doa:  2010-02-24, 2013-03-25, 2013-07-29, 2013-10-27"
## [1] "pulse:  69, 75, 80, 84"
## [1] "temp:  98.49, 98.82, 98.74, 98.54"
## [1] "fluid:  0.81, 0.59, 0.28, 0.4"
##
## [1] "name:  Yol"
## [1] "gender: male "
## [1] "dob:  2026-03-11"
## [1] "doa:  2010-02-22, 2011-12-27, 2012-03-10, 2012-11-26, 2013-03-24, 2014-01-28"
## [1] "pulse:  69, 78, 87, 84, 89, 92"
## [1] "temp:  98.29, 98.44, 98.78, 98.87, 98.27, 98.26"
## [1] "fluid:  0.03, 0.13, 0.12, 0.39, 0.97, 0.14"
##
## [1] "name:  Yak"
## [1] "gender: female "
## [1] "dob:  2023-09-15"
## [1] "doa:  2011-07-19, 2012-04-07, 2012-07-11, 2012-08-30"
## [1] "pulse:  90, 88, 75, 81"
## [1] "temp:  98.58, 97.53, 98.58, 99.11"
## [1] "fluid:  0.26, 0.29, 0.6, 0.66"
##
## [1] "name:  Gaf"
## [1] "gender: female "
## [1] "dob:  2018-04-27"
## [1] "doa:  2010-07-19, 2011-05-03, 2012-04-24, 2012-08-06, 2013-08-21"
## [1] "pulse:  89, 91, 77, 75, 90"
## [1] "temp:  98.32, 98.01, 98.96, 98.52, 98.61"
## [1] "fluid:  0.42, 0.47, 0.74, 0.62, 0.36"
##
## [1] "name:  Kuw"
## [1] "gender: female "
## [1] "dob:  2020-11-07"
## [1] "doa:  2010-10-03, 2010-10-29, 2011-09-16, 2012-07-10"
```

```
## [1] "pulse:  72, 81, 71, 82"
## [1] "temp:  98.21, 98.17, 98.65, 98.49"
## [1] "fluid:  0.29, 0.93, 0.25, 0.12"
##
## [1] "name:  Mav"
## [1] "gender: female "
## [1] "dob:  2029-07-16"
## [1] "doa:  2010-02-08, 2010-04-19, 2010-06-11, 2012-03-02"
## [1] "pulse:  63, 83, 66, 88"
## [1] "temp:  99.07, 98.45, 97.95, 98"
## [1] "fluid:  0.01, 0.79, 0.79, 0.5"
##
## [1] "name:  Fel"
## [1] "gender: male "
## [1] "dob:  2025-08-16"
## [1] "doa:  2010-09-26, 2012-06-24"
## [1] "pulse:  65, 81"
## [1] "temp:  98.21, 98.51"
## [1] "fluid:  0.06, 0.24"
##
## [1] "name:  Say"
## [1] "gender: female "
## [1] "dob:  2014-09-22"
## [1] "doa:  2010-03-14"
## [1] "pulse:  "
## [1] "temp:  .5"
## [1] "fluid:  1"
```