Bios 6301: Assignment 8

Catherine Greene

Due Tuesday, 15 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

```
install.packages("readxl", repos="http://cran.us.r-project.org") # it doesn't like my mirror
```

```
##
## The downloaded binary packages are in
##  /var/folders/fn/3nmxs93x7wn7qxpwl0ky75t00000gn/T//RtmpB9LnOt/downloaded_packages
```

```
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='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
```

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

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

```
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
```

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

```
#print(dat) # when I included this my document was ~500 pgs
# tbl_df
# uses tbl_df because that class is listed first in the vector when I searched for class(dat), and it h
```

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

```
class(dat) <- "data.frame"
```

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

```
class(dat)
```

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

```
#print(dat)
# print.data.frame is the method applied since it's just a data.frame object now
```

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.default <- 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) {
  counts <- c()
  for (i in seq(ncol(x))) {
    counts <- c(counts, length(unique(x[,i])))
  }
  counts
}
```

9. Check your function (2 points)

```
nUnique(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)
p <- makePatient() # I think this is what the instructions were saying, but I'd like to be able to lear

patient_record <- list(name='Yes', gender='male', date_of_birth='1977-05-03', date_of_admission=c('2013
class(patient_record) <- 'medicalRecord'
attributes(patient_record)
```

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

```
print.medicalRecord <- function(record) {
  cat(sprintf("name: %s\ngender: %s\ndate_of_birth: %s\ndate_of_admission: %s\npulse: %s\ntemperature: %
              record$name, record$gender, record$date_of_birth, paste(record$date_of_admission, collaps
}
```

```
methods(class='medicalRecord')
```

```
## [1] print
## see '?methods' for accessing help and source code
```

```
# yay! print shows up

suppressWarnings(print.medicalRecord(patient_record)) # it doesn't like that I have multiple arguments
```

```
## name: Yes
## gender: male
## date_of_birth: 1977-05-03
## date_of_admission: 2013-06-09 2013-07-02
## pulse: 79 78
## temperature: 98.07 97.5
## fluid_intake: 0.28 0.52
```

```
class(patient_record)
```

```
## [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
mean.medicalRecord <- function(record) {
  cat(sprintf("mean pulse: %s\nmean temperature: %s\nmean fluids: %s\n",
              mean(record$pulse), mean(record$temperature), mean(record$fluid_intake), "\n"))
}

suppressWarnings(mean.medicalRecord(patient_record))
```

```
## mean pulse: 78.5
## mean temperature: 97.785
## mean fluids: 0.4
```

```
# I kinda already did this up above, so here's a modified print method for medicalRecord
# be careful if sorting by date - need to also sort the corresponding measurements
print.medicalRecord.date <- function(record) {
  df <- data.frame(record$date_of_admission, # if there are multiple measurements, arrange by date
                   record$pulse,
                   record$temperature,
                   record$fluid_intake
                  )
  sorted_df <- df[order(df[,1], decreasing=FALSE), ] # sort by the date column, then extract values bac
  cat(sprintf("name: %s\ngender: %s\ndate_of_birth: %s\ndate_of_admission: %s\npulse: %s\ntemperature: %
              record$name, record$gender, record$date_of_birth, paste(sorted_df[,1], collapse=" "), pas
}

suppressWarnings(print.medicalRecord.date(patient_record))
```

```
## name: Yes
## gender: male
## date_of_birth: 1977-05-03
## date_of_admission: 2013-06-09 2013-07-02
```

```
## pulse: 79 78
## temperature: 98.07 97.5
## fluid_intake: 0.28 0.52
```

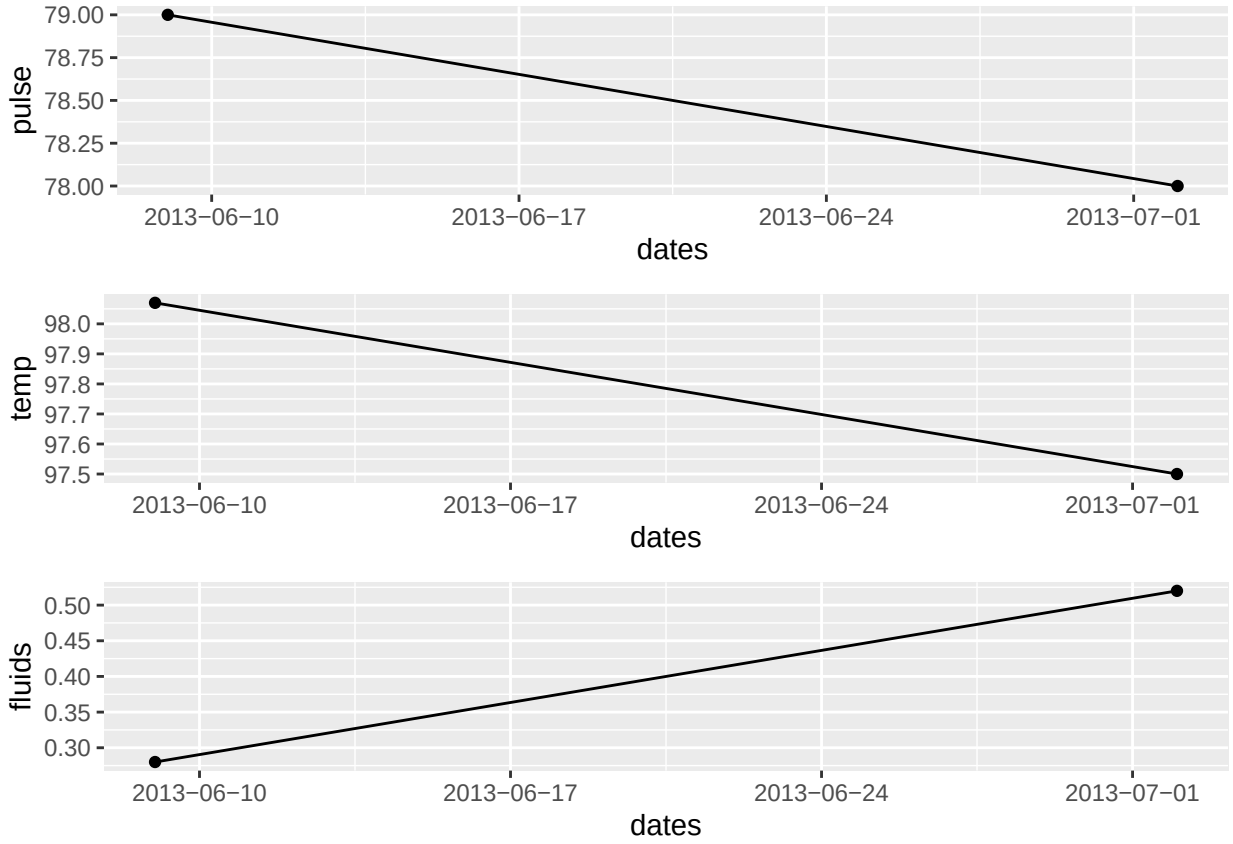
```
install.packages('gridExtra')
```

```
## Error in contrib.url(repos, "source"): trying to use CRAN without setting a mirror
```

```
library(ggplot2)
library('gridExtra')

plot.medicalRecord <- function(record) {
  df <- data.frame(record$date_of_admission, # I want to keep this data frame for plotting
                   record$pulse,
                   record$temperature,
                   record$fluid_intake
                  )
  sorted_df <- df[order(df[,1], decreasing=FALSE), ]
  dates=as.Date(sorted_df[,1])
  pulse=sorted_df[,2]
  temp=sorted_df[,3]
  fluids=sorted_df[,4]
  # I want three plots in one window, since each measurement is on a different scale
  # not perfect, but I couldn't combine them because of the different y axes
  a <- ggplot(sorted_df, aes(x=dates, y=pulse)) + geom_line() + geom_point() + scale_x_date(date_labels
  b <- ggplot(sorted_df, aes(x=dates, y=temp)) + geom_line() + geom_point() + scale_x_date(date_labels =
  c <- ggplot(sorted_df, aes(x=dates, y=fluids)) + geom_line() + geom_point() + scale_x_date(date_labels
  grid.arrange(a, b, c, nrow=3)
}

plot.medicalRecord(patient_record)
```



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 <- replicate(10, makePatient(), simplify=FALSE) # 10 patients
class(cohort) <- c('cohortRecord', 'medicalRecord')
inherits(cohort, 'medicalRecord')
```

```
## [1] TRUE
```

```
#cohort[[1]] # first person in the cohort
#cohort[[1]][1] # name
#cohort[[1]][2] # gender
#cohort[[1]][3] # date_of_birth
#cohort[[1]][4] # date_of_admission
#cohort[[1]][5] # pulse
#cohort[[1]][6] # temperature
#cohort[[1]][7] # fluid_intake

for (i in 1:(length(cohort))) { # cohort is in groups of 7 (per individual)
  names(cohort[[i]]) <- c('name', 'gender', 'date_of_birth', 'date_of_admission', 'pulse', 'temperature
}

# don't tell on me, pretty sure this would violate HIPAA
mean.cohort <- function(record) { # where record is the overall cohort
  for (i in 1:(length(record))) {
```

```
    cat(record[[i]]$name, '\n')
    suppressWarnings(mean.medicalRecord(record[[i]]))
    cat('\n') # skips a line so you can look at each patient's average info a little more easily
  }
}
mean.cohort(cohort)
```

```
## Yes
## mean pulse: 78.5
## mean temperature: 97.785
## mean fluids: 0.4
##
## Fal
## mean pulse: 86.3333333333333
## mean temperature: 98.3966666666667
## mean fluids: 0.413333333333333
##
## Zog
## mean pulse: 77
## mean temperature: 98.6475
## mean fluids: 0.52
##
## Yol
## mean pulse: 83.1666666666667
## mean temperature: 98.485
## mean fluids: 0.296666666666667
##
## Yak
## mean pulse: 83.5
## mean temperature: 98.45
## mean fluids: 0.4525
##
## Gaf
## mean pulse: 84.4
## mean temperature: 98.484
## mean fluids: 0.522
##
## Kuw
## mean pulse: 76.5
## mean temperature: 98.38
## mean fluids: 0.3975
##
## Mav
## mean pulse: 75
## mean temperature: 98.3675
## mean fluids: 0.5225
##
## Fel
## mean pulse: 73
## mean temperature: 98.36
## mean fluids: 0.15
##
## Say
## mean pulse: 77
```

```
## mean temperature: 98.54
## mean fluids: 0.15
```

```
print.cohort <- function(record) {
  for (i in 1:(length(record))) {
    suppressWarnings(print.medicalRecord.date(record[[i]]))
    cat('\n\n')
  }
}
print.cohort(cohort)
```

```
## name: Yes
## gender: male
## date_of_birth: 1977-05-03
## date_of_admission: 2013-06-09 2013-07-02
## pulse: 79 78
## temperature: 98.07 97.5
## fluid_intake: 0.28 0.52
##
## name: Fal
## gender: male
## date_of_birth: 1988-05-24
## date_of_admission: 2010-11-16 2013-03-24 2013-09-12
## pulse: 76 87 96
## temperature: 98.23 98.21 98.75
## fluid_intake: 0.18 0.1 0.96
##
## name: Zog
## gender: male
## date_of_birth: 1988-12-14
## date_of_admission: 2010-02-24 2013-03-25 2013-07-29 2013-10-27
## pulse: 84 69 75 80
## temperature: 98.54 98.49 98.82 98.74
## fluid_intake: 0.4 0.81 0.59 0.28
##
## name: Yol
## gender: male
## date_of_birth: 1986-03-11
## date_of_admission: 2010-02-22 2011-12-27 2012-03-10 2012-11-26 2013-03-24 2014-01-28
## pulse: 84 89 87 92 78 69
## temperature: 98.87 98.27 98.78 98.26 98.44 98.29
## fluid_intake: 0.39 0.97 0.12 0.14 0.13 0.03
##
## name: Yak
## gender: female
## date_of_birth: 1983-09-15
## date_of_admission: 2011-07-19 2012-04-07 2012-07-11 2012-08-30
## pulse: 75 88 81 90
## temperature: 98.58 97.53 99.11 98.58
## fluid_intake: 0.6 0.29 0.66 0.26
##
## name: Gaf
## gender: female
## date_of_birth: 1978-04-27
## date_of_admission: 2010-07-19 2011-05-03 2012-04-24 2012-08-06 2013-08-21
```

```
## pulse: 91 90 89 77 75
## temperature: 98.01 98.61 98.32 98.96 98.52
## fluid_intake: 0.47 0.36 0.42 0.74 0.62
##
## name: Kuw
## gender: female
## date_of_birth: 1980-11-07
## date_of_admission: 2010-10-03 2010-10-29 2011-09-16 2012-07-10
## pulse: 82 81 72 71
## temperature: 98.49 98.17 98.21 98.65
## fluid_intake: 0.12 0.93 0.29 0.25
##
## name: Mav
## gender: female
## date_of_birth: 1989-07-16
## date_of_admission: 2010-02-08 2010-04-19 2010-06-11 2012-03-02
## pulse: 66 88 83 63
## temperature: 97.95 98 98.45 99.07
## fluid_intake: 0.79 0.5 0.79 0.01
##
## name: Fel
## gender: male
## date_of_birth: 1985-08-16
## date_of_admission: 2010-09-26 2012-06-24
## pulse: 81 65
## temperature: 98.51 98.21
## fluid_intake: 0.24 0.06
##
## name: Say
## gender: female
## date_of_birth: 1974-09-22
## date_of_admission: 2010-03-14
## pulse: 77
## temperature: 98.54
## fluid_intake: 0.15
```