Bios 6301: Assignment 8

Haley Yaremych

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)

```
library(readxl)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
tryCatch(methods(,tbl_df), error = function(e) e)
```

```
##  [1] [             [[            [[<-          [<-           $
##  [6] $<-           arrange_      as.data.frame coerce        distinct_
## [11] filter_       group_data    initialize    mutate_       names<-
## [16] Ops           row.names<-   show          slice_        slotsFromS3
## [21] str           summarise_    tbl_sum
## see '?methods' for accessing help and source code
```

```
tryCatch(methods(,tbl), error = function(e) e)
```

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

```
tryCatch(methods(,data.frame), error = function(e) e)
```

```
##   [1] [                 [[                [[<-              [<-
##   [5] $<-               aggregate         anti_join         anyDuplicated
##   [9] anyNA             arrange_          arrange           as_tibble
##  [13] as.data.frame     as.list           as.matrix         as.tbl
##  [17] auto_copy         by                cbind             coerce
##  [21] collapse          collect           compute           count
##  [25] dim               dimnames          dimnames<-        distinct_
##  [29] distinct          do_               do                dplyr_col_modify
##  [33] dplyr_reconstruct dplyr_row_slice   droplevels        duplicated
##  [37] edit              filter_           filter            format
##  [41] formula           full_join         glimpse           group_by_
##  [45] group_by          group_data        group_indices_    group_indices
##  [49] group_keys        group_map         group_modify      group_nest
##  [53] group_size        group_split       group_trim        group_vars
##  [57] groups            head              initialize        inner_join
##  [61] intersect         is.na             left_join         Math
##  [65] merge             mutate_           mutate            n_groups
##  [69] na.exclude        na.omit           nest_by           nest_join
##  [73] Ops               plot              print             prompt
##  [77] pull              rbind             relocate          rename_
##  [81] rename_with       rename            right_join        row.names
##  [85] row.names<-       rows_delete       rows_insert       rows_patch
##  [89] rows_update       rows_upsert       rowsum            rowwise
##  [93] same_src          sample_frac       sample_n          select_
##  [97] select            semi_join         setdiff           setequal
## [101] show              slice_            slice_head        slice_max
## [105] slice_min         slice_sample      slice_tail        slice
## [109] slotsFromS3       split             split<-           stack
## [113] str               subset            summarise_        summarise
## [117] summary           Summary           t                 tail
## [121] tally             tbl_vars          transform         transmute_
## [125] transmute         type.convert      ungroup           union_all
## [129] union             unique            unstack           within
## 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 being dispatched is a print function specifically for objects of the `tbl_df` class.

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

```
dat = as.data.frame(dat)
```

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

Now, since `dat` is a type data.frame, the print method being dispatched is the print method that is specifically for objects of type `data.frame`, which could also be called with `print.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.default = function(x){
  return(length(unique(x)))
}
methods("nUnique")
```

```
## [1] nUnique.default
## see '?methods' for accessing help and source code
```

7. Check your function (2 points)

```
nUnique(letters) # should return 26
```

```
## [1] 26
```

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

```
## [1] 10
```

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(df){
  apply(df, MARGIN=2, FUN=nUnique)
}
```

9. Check your function (2 points)

```
nUnique(dat)
```

```
##     Procedure Code Category           ICD-10-PCS Codes
##                          39                       9697
## Procedure Code Descriptions                Code Status
##                        9697                          4
```

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)

medicalRecord = function(x){
  class(x) = 'medicalRecord'
  attr(x, 'name') = x[[1]]
  attr(x, 'gender') = as.character(x[[2]])
  attr(x, 'date_of_birth') = x[[3]]
  attr(x, 'date_of_admission') = x[[4]]
  attr(x, 'pulse') = x[[5]]
  attr(x, 'temp') = x[[6]]
  attr(x, 'fluid_intake') = x[[7]]
  return(x)
}

rec = medicalRecord(makePatient())
print(rec)
```

```
## [[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"
## attr(,"name")
## [1] "Yes"
## attr(,"gender")
## [1] "male"
## attr(,"date_of_birth")
## [1] "1977-05-03"
## attr(,"date_of_admission")
## [1] "2013-06-09" "2013-07-02"
## attr(,"pulse")
## [1] 79 78
## attr(,"temp")
## [1] 98.07 97.50
## attr(,"fluid_intake")
## [1] 0.28 0.52
```

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)

```
library(ggplot2)

mean.medicalRecord = function(x){
  ret = rep(0, 3)
    ret[1] = mean(attr(x, 'pulse'))
    ret[2] = mean(attr(x, 'temp'))
    ret[3] = mean(attr(x, 'fluid_intake'))
  return(ret)
}

print.medicalRecord = function(x){
  atts = attributes(x)[-1]

  DOB = as.character(attr(x, 'date_of_birth'), format="%Y-%m-%d")

  DOA = attr(x, 'date_of_admission')
  for (i in 1:length(DOA)){
    DOA[i] = as.character(DOA[i], format="%Y-%m-%d")
  }
  DOA = paste(sort(DOA))

  df = data.frame(Attribute = c("Name:", "Gender:", "DOB:",
                                "Pulse:", "Temp:", "Fluid Intake:"),
                  Value=rep(0,length(atts)-1))

  df$Value[1] = attr(x, 'name')
  df$Value[2] = attr(x, 'gender')
  df$Value[3] = DOB
  df$Value[4] = list(attr(x, 'pulse'))
  df$Value[5] = list(attr(x, 'temp'))
  df$Value[6] = list(attr(x, 'fluid_intake'))

  print(df)
  cat("Admission Date(s): ")
```

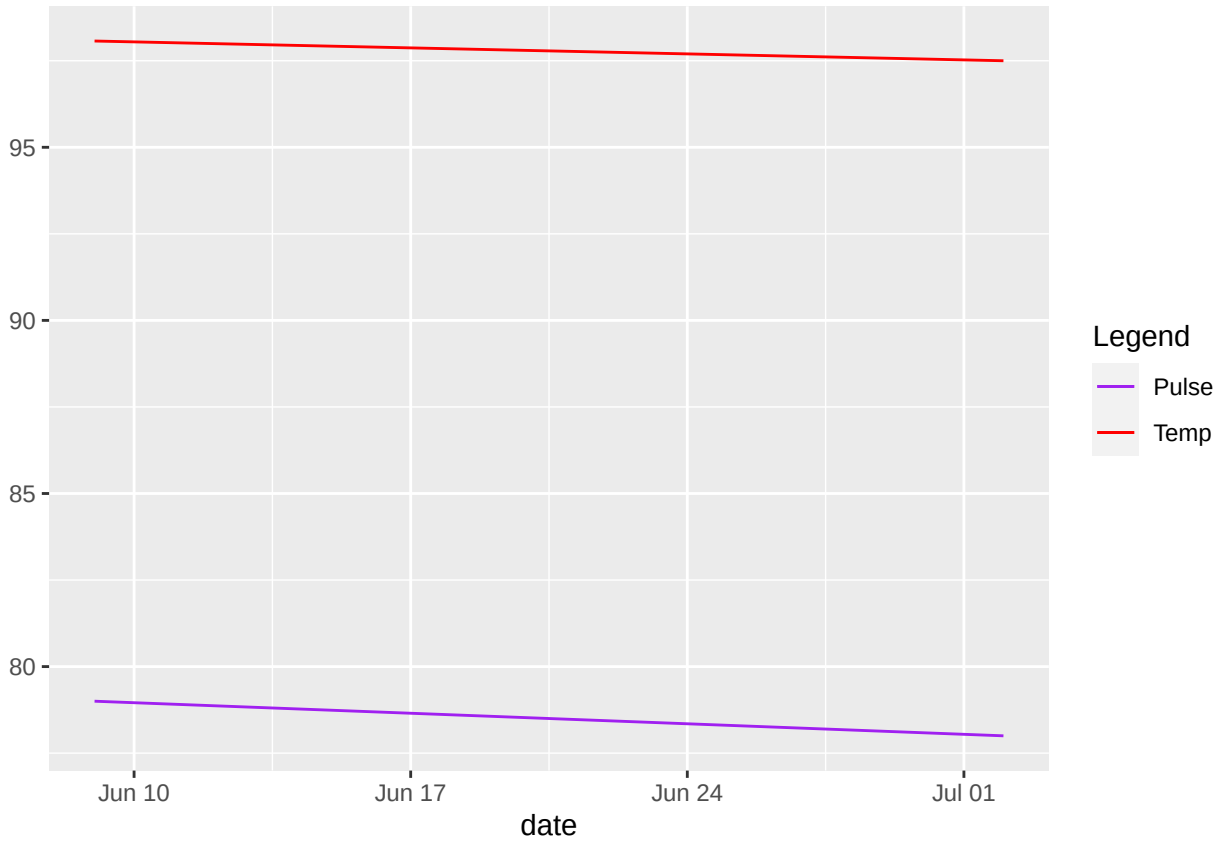
```
  cat(DOA)
}

plot.medicalRecord = function(x){
  df = data.frame(date = attr(x, 'date_of_admission'),
                  temp = attr(x, 'temp'),
                  pulse = attr(x, 'pulse'))
  colors <- c("Temp" = "red", "Pulse" = "purple")
  ggplot(data=df, aes(x=date)) +
     geom_line(aes(y=temp, color="Temp")) +
     geom_line(aes(y=pulse, color="Pulse")) +
     labs(color="Legend") +
     scale_color_manual(values=colors) +
    theme(axis.title.y = element_blank())
}

mean(rec)
```

```
## [1] 78.500 97.785  0.400
```

```
print(rec)
```

```
##       Attribute        Value
## 1         Name:          Yes
## 2       Gender:         male
## 3          DOB:   1977-05-03
## 4        Pulse:       79, 78
## 5         Temp: 98.07, 97.50
## 6 Fluid Intake:   0.28, 0.52
## Admission Date(s): 2013-06-09 2013-07-02
```

```
plot(rec)
```



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)

# accepts a list of medicalRecords
cohort = function(x){
  new = vector(mode = "list", length = length(x))

  for (i in 1:length(x)){
    new[[i]] = x[[i]]
  }
  class(new) = 'cohort'
  return(new)
}

mean.cohort = function(x){
  t(sapply(x, FUN=mean))
}

print.cohort = function(x){
  for (i in 1:length(x)){
    print(x[[i]])
    cat("\n--------\n")
  }
}
```

```r
pats <- vector(mode = "list", length = 10)
for (i in 1:10){
  pats[[i]] = medicalRecord(makePatient())
}

cohort = cohort(pats)

mean(cohort)
```

```
##           [,1]     [,2]      [,3]
##  [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
```

```r
print(cohort)
```

```
##       Attribute        Value
## 1         Name:          Yes
## 2       Gender:         male
## 3          DOB:   1977-05-03
## 4        Pulse:       79, 78
## 5         Temp: 98.07, 97.50
## 6 Fluid Intake:   0.28, 0.52
## Admission Date(s): 2013-06-09 2013-07-02
## --------
##       Attribute               Value
## 1         Name:                 Fal
## 2       Gender:                male
## 3          DOB:          1988-05-24
## 4        Pulse:          76, 96, 87
## 5         Temp: 98.23, 98.75, 98.21
## 6 Fluid Intake:    0.18, 0.96, 0.10
## Admission Date(s): 2010-11-16 2013-03-24 2013-09-12
## --------
##       Attribute                      Value
## 1         Name:                        Zog
## 2       Gender:                       male
## 3          DOB:                1988-12-14
## 4        Pulse:             69, 75, 80, 84
## 5         Temp: 98.49, 98.82, 98.74, 98.54
## 6 Fluid Intake:     0.81, 0.59, 0.28, 0.40
## Admission Date(s): 2010-02-24 2013-03-25 2013-07-29 2013-10-27
## --------
##       Attribute                                    Value
## 1         Name:                                      Yol
## 2       Gender:                                     male
## 3          DOB:                                1986-03-11
```

```
## 4        Pulse:                   69, 78, 87, 84, 89, 92
## 5         Temp: 98.29, 98.44, 98.78, 98.87, 98.27, 98.26
## 6 Fluid Intake:       0.03, 0.13, 0.12, 0.39, 0.97, 0.14
## Admission Date(s): 2010-02-22 2011-12-27 2012-03-10 2012-11-26 2013-03-24 2014-01-28
## --------
##       Attribute                      Value
## 1         Name:                        Yak
## 2       Gender:                     female
## 3          DOB:                  1983-09-15
## 4        Pulse:              90, 88, 75, 81
## 5         Temp: 98.58, 97.53, 98.58, 99.11
## 6 Fluid Intake:     0.26, 0.29, 0.60, 0.66
## Admission Date(s): 2011-07-19 2012-04-07 2012-07-11 2012-08-30
## --------
##       Attribute                               Value
## 1         Name:                                 Gaf
## 2       Gender:                              female
## 3          DOB:                           1978-04-27
## 4        Pulse:                 89, 91, 77, 75, 90
## 5         Temp: 98.32, 98.01, 98.96, 98.52, 98.61
## 6 Fluid Intake:      0.42, 0.47, 0.74, 0.62, 0.36
## Admission Date(s): 2010-07-19 2011-05-03 2012-04-24 2012-08-06 2013-08-21
## --------
##       Attribute                      Value
## 1         Name:                        Kuw
## 2       Gender:                     female
## 3          DOB:                  1980-11-07
## 4        Pulse:              72, 81, 71, 82
## 5         Temp: 98.21, 98.17, 98.65, 98.49
## 6 Fluid Intake:     0.29, 0.93, 0.25, 0.12
## Admission Date(s): 2010-10-03 2010-10-29 2011-09-16 2012-07-10
## --------
##       Attribute                      Value
## 1         Name:                        Mav
## 2       Gender:                     female
## 3          DOB:                  1989-07-16
## 4        Pulse:              63, 83, 66, 88
## 5         Temp: 99.07, 98.45, 97.95, 98.00
## 6 Fluid Intake:     0.01, 0.79, 0.79, 0.50
## Admission Date(s): 2010-02-08 2010-04-19 2010-06-11 2012-03-02
## --------
##       Attribute        Value
## 1         Name:          Fel
## 2       Gender:         male
## 3          DOB:   1985-08-16
## 4        Pulse:       65, 81
## 5         Temp: 98.21, 98.51
## 6 Fluid Intake:   0.06, 0.24
## Admission Date(s): 2010-09-26 2012-06-24
## --------
##       Attribute      Value
## 1         Name:        Say
## 2       Gender:     female
## 3          DOB: 1974-09-22
```

```
## 4        Pulse:         77
## 5         Temp:      98.54
## 6 Fluid Intake:       0.15
## Admission Date(s): 2010-03-14
## --------
```