BIOS6301: Homework 5

Sarah Lotspeich

11/1/2016

Question 1

24 points

Import the HAART dataset (haart.csv) from the GitHub repository into R, and perform the following manipulations: (4 points each)

```
library(lubridate)
haart <- read.csv("https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/haart.csv")
```

1. Convert date columns into a usable (for analysis) format. Use the table command to display the counts of the year from init.date.

```
# convert date columns
haart[, "init.date"] <- as.POSIXct(haart[, "init.date"], format = "%m/%d/%y")
haart[, "last.visit"] <- as.POSIXct(haart[, "last.visit"], format = "%m/%d/%y")
haart[, "date.death"] <- as.POSIXct(haart[, "date.death"], format = "%m/%d/%y")

# display counts of the year from init.date
table(year(haart[, "init.date"]))
```

```
##
## 1998 2000 2001 2002 2003 2004 2005 2006 2007
##    1    5   17   60  270  292  207  104   44
```

1.2. Create an indicator variable (one which takes the values 0 or 1 only) to represent death within 1 year of the initial visit. How many observations died in year 1?

```
# create indicator variable to represent death within 1 year of the initial
# visit
haart$death.within.year <- (difftime(haart$date.death, haart$last.visit, units = "days") <=
    365)
haart$death.within.year[is.na(haart$death.within.year)] <- 0

# calculate number of paricipants who died in year 1
(yr1.deaths <- sum(haart$death.within.year))
```

```
## [1] 104
```

1.3. Use the init.date, last.visit and death.date columns to calculate a followup time (in days), which is the difference between the first and either the last visit or a death event (whichever comes first).

```r
#function that takes who dataframe, adds a followup time column and calculates values, then returns the
calculate.followup <- function(dataframe)
{
  for (i in 1:nrow(dataframe))
  {
    if (dataframe$death[i] == 1) #check to see if death came first
    {
      dataframe$followup.time[i] <- difftime(dataframe$date.death[i],dataframe$init.date[i],units="days
    }
    else
    {
      dataframe$followup.time[i] <- difftime(dataframe$last.visit[i],dataframe$init.date[i],units="days
    }
  }
  return(dataframe)
}

haart <- calculate.followup(haart)
```

1.4. If these times are longer than 1 year, censor them (this means if the value is above 365, set followup to 365).

```r
#function that takes in a vector of followup.times and censors them so that 365 is the maximum, then re
censor.followup <- function(followup.time)
{
  for (i in 1:length(followup.time))
  {
    if (followup.time[i] > 365)
    {
      followup.time[i] <- 365
    }
  }
  return(followup.time)
}

haart$followup.time <- censor.followup(haart$followup.time)
```

1.5. Print the quantile for this new variable.

```r
quantile(haart$followup.time)
```

```
##    0%   25%   50%   75%  100%
##   0.0 329.5 365.0 365.0 365.0
```

1.6. Create another indicator variable representing loss to followup; this means the observation is not known to be dead but does not have any followup visits after the first year. How many records are lost-to-followup?

```r
#function that takes in whole dataframe and appens an indicator column for whether or not individuals w
calculate.losstofollowup <- function(dataframe)
{
  for (i in 1:nrow(dataframe))
  {
```

```
    if (dataframe$death[i] == 1)#if dead, then they weren't "lost-to-followup"
    {
      dataframe$loss.to.followup[i] <- 0
    }
    else
    {
      if (as.integer(difftime(dataframe$last.visit[i],dataframe$init.date[i],units="days")) <= 365) #if
      {
        dataframe$loss.to.followup[i] <- 1
      }
      else dataframe$loss.to.followup[i] <- 0 #if not dead, and they have had a followup after the firs
    }
  }
  return(dataframe)
}

#add lost-to-followup column to the haart dataframe
haart <- calculate.losstofollowup(haart)

sum(haart$loss.to.followup) #sum of indicator variables gives number of patients "lost-to-followup"
```

```
## [1] 173
```

So, from this we can see that 173 records were lost-to-followup.

1.7. Recall our work in class, which separated the init.reg field into a set of indicator variables, one for each unique drug. Create these fields and append them to the database as new columns.

```
create.regimens <- function(dataframe) {
    init.reg <- as.character(dataframe[, "init.reg"])
    dataframe[["init.reg_list"]] <- strsplit(init.reg, ",")
    unique.drugs <- unique(unlist(dataframe$init.reg_list))
    reg.drugs <- matrix(FALSE, nrow = nrow(dataframe), ncol = length(unique.drugs))
    for (i in seq_along(unique.drugs)) {
        reg.drugs[, i] <- sapply(dataframe$init.reg_list, function(x) unique.drugs[i] %in%
            x)
    }
    reg.drugs <- data.frame(reg.drugs)
    names(reg.drugs) <- unique.drugs
    dataframe <- cbind(dataframe, reg.drugs)
    return(dataframe)
}

haart <- create.regimens(haart)
```

1.8. Which drug regimen are found over 100 times?

```
regimen <- matrix(nrow = nrow(haart), ncol = 1)
for (j in 17:34) {
    for (i in 1:nrow(haart)) {
        if (haart[i, j] == TRUE) {
            if (j == 17) {
                regimen[i] <- colnames(haart)[j]
```

```r
            } else {
                regimen[i] <- paste(regimen[i], colnames(haart)[j])
            }
        }
    }
}
haart <- cbind(haart, regimen)
sort(table(regimen))
```

```
## regimen
##        3TC ABC IDV RTV           3TC ABC RTV    3TC AZT ABC LPV RTV
##                      1                     1                      1
##    3TC AZT ABC RTV SQV           3TC AZT DDI        3TC AZT EFV NFV
##                      1                     1                      1
##        3TC AZT RTV FPV           3TC EFV TDF        3TC LPV RTV TDF
##                      1                     1                      1
##        3TC RTV TDF FPV    NA ABC DDI LPV RTV     NA ABC DDI RTV ATV
##                      1                     1                      1
##     NA D4T ABC LPV RTV    NA D4T ABC RTV SQV         NA D4T RTV SQV
##                      1                     1                      1
## NA DDI LPV RTV SQV T20        NA EFV D4T DDC         NA EFV DDI FTC
##                      1                     1                      1
##         NA NVP FTC TDF    NA RTV FTC TDF ATV        3TC D4T LPV RTV
##                      1                     1                      2
##            3TC NVP ABC        NA AZT EFV DDI         NA EFV D4T ABC
##                      2                     2                      2
##     NA LPV RTV FTC TDF        NA NVP D4T DDI         NA NVP LPV RTV
##                      2                     2                      2
##            3TC D4T NFV        NA EFV FTC TDF        3TC ABC RTV SQV
##                      3                     3                      4
##            3TC AZT NFV       3TC DDI LPV RTV         NA EFV D4T DDI
##                      4                     4                      4
##        3TC D4T IDV RTV           3TC NVP DDI        3TC AZT IDV RTV
##                      6                     6                      8
##        3TC D4T RTV SQV           3TC EFV ABC            3TC AZT IDV
##                      8                    11                     12
##        3TC AZT RTV SQV           3TC EFV DDI        3TC AZT LPV RTV
##                     13                    15                     16
##            3TC AZT ABC           3TC EFV D4T            3TC NVP D4T
##                     29                    54                     61
##            3TC AZT NVP           3TC AZT EFV
##                    284                   421
```

From this, we can see that the only regimens that were prescribed more than 100 times were "3TC AZT NVP" and "3TC AZT EFV".

Turning this into a data frame is as simple as a call to data.frame, using all_drugs as a set of column labels:

1.9. The dataset haart2.csv contains a few additional observations for the same study. Import these and append them to your master dataset (if you were smart about how you coded the previous steps, cleaning the additional observations should be easy!). Show the first five records and the last five records of the complete (and clean) data set.

```r
haart <- data.frame(read.csv("https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/haa
haart2 <- data.frame(read.csv("https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/ha
haart.merged <- merge(haart, haart2, all = TRUE)

# convert date columns
haart.merged[, "init.date"] <- as.POSIXct(haart.merged[, "init.date"], format = "%m/%d/%y")
haart.merged[, "last.visit"] <- as.POSIXct(haart.merged[, "last.visit"], format = "%m/%d/%y")
haart.merged[, "date.death"] <- as.POSIXct(haart.merged[, "date.death"], format = "%m/%d/%y")

# create indicator variable to represent death within 1 year of the initial
# visit
haart.merged$death.within.year <- (difftime(haart.merged$date.death, haart.merged$last.visit,
    units = "days") <= 365)
haart.merged$death.within.year[is.na(haart.merged$death.within.year)] <- 0

# create followup time field
haart.merged <- calculate.followup(haart.merged)

# censor followup time
haart.merged$followup.time <- censor.followup(haart.merged$followup.time)

# add indicator for loss-to-followup
haart.merged <- calculate.losstofollowup(haart.merged)

# add indicators for regimen
haart.merged <- create.regimens(haart.merged)

regimen <- matrix(nrow = nrow(haart.merged), ncol = 1)
for (j in 17:34) {
    for (i in 1:nrow(haart.merged)) {
        if (haart.merged[i, j] == TRUE) {
            if (j == 17) {
                regimen[i] <- colnames(haart.merged)[j]
            } else {
                regimen[i] <- paste(regimen[i], colnames(haart.merged)[j])
            }
        }
    }
}
haart.merged <- cbind(haart.merged, regimen)

haart.merged[1:5, ]  #first 5
```

```
##   male age aids cd4baseline    logvl weight hemoglobin    init.reg
## 1    0  18    0          89 5.184231     NA         NA 3TC,AZT,EFV
## 2    0  18    0         280       NA 52.164         11 3TC,AZT,EFV
## 3    0  18    0         431 5.342423 58.000         NA 3TC,AZT,NVP
## 4    0  19    0          51 5.618615 48.600         NA 3TC,AZT,NVP
## 5    0  19    0         180 4.121330     NA         NA 3TC,AZT,NVP
##    init.date last.visit death date.death death.within.year followup.time
## 1 2003-11-03 2006-04-12     0       <NA>                 0           365
## 2 2004-02-19 2008-03-14     0       <NA>                 0           365
## 3 2007-03-13 2007-03-13     0       <NA>                 0             0
```

```
## 4 2005-12-07 2007-04-17     0       <NA>                 0           365
## 5 2006-09-08 2006-10-15     0       <NA>                 0            37
##   loss.to.followup init.reg_list  3TC  AZT   EFV   NVP   NFV   ABC   LPV
## 1                0 3TC, AZT, EFV TRUE TRUE  TRUE FALSE FALSE FALSE FALSE
## 2                0 3TC, AZT, EFV TRUE TRUE  TRUE FALSE FALSE FALSE FALSE
## 3                1 3TC, AZT, NVP TRUE TRUE FALSE  TRUE FALSE FALSE FALSE
## 4                0 3TC, AZT, NVP TRUE TRUE FALSE  TRUE FALSE FALSE FALSE
## 5                1 3TC, AZT, NVP TRUE TRUE FALSE  TRUE FALSE FALSE FALSE
##     RTV   D4T   DDI   IDV   SQV   T20   FPV   TDF   ATV   FTC   DDC
## 1 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 2 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 3 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 4 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 5 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##       regimen
## 1 3TC AZT EFV
## 2 3TC AZT EFV
## 3 3TC AZT NVP
## 4 3TC AZT NVP
## 5 3TC AZT NVP
```

```
haart.merged[1000:1004, ]  #last 5
```

```
##      male age aids cd4baseline   logvl  weight hemoglobin        init.reg
## 1000    1  66    0         298 4.09496      NA         NA     3TC,AZT,EFV
## 1001    1  67    0          95      NA 66.6792         16     3TC,AZT,EFV
## 1002    1  69    0          NA      NA      NA         NA 3TC,AZT,RTV,SQV
## 1003    1  80    0         267      NA 53.0712         NA     3TC,AZT,NVP
## 1004    1  89    0           9      NA 43.5456         10     3TC,ABC,AZT
##       init.date last.visit death date.death death.within.year
## 1000 2006-06-08 2007-02-12     0       <NA>                 0
## 1001 2004-02-13 2008-02-21     0       <NA>                 0
## 1002 2006-04-01 2007-09-13     0       <NA>                 0
## 1003 2004-11-08 2006-11-20     1 2006-11-26                 1
## 1004 2004-12-15 2006-04-11     0       <NA>                 0
##      followup.time loss.to.followup      init.reg_list  3TC  AZT   EFV
## 1000      249.0417                1      3TC, AZT, EFV TRUE TRUE  TRUE
## 1001      365.0000                0      3TC, AZT, EFV TRUE TRUE  TRUE
## 1002      365.0000                0 3TC, AZT, RTV, SQV TRUE TRUE FALSE
## 1003      365.0000                0      3TC, AZT, NVP TRUE TRUE FALSE
## 1004      365.0000                0      3TC, ABC, AZT TRUE TRUE FALSE
##        NVP   NFV   ABC   LPV   RTV   D4T   DDI   IDV   SQV   T20   FPV
## 1000 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1001 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1002 FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
## 1003  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1004 FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##        TDF   ATV   FTC   DDC         regimen
## 1000 FALSE FALSE FALSE FALSE     3TC AZT EFV
## 1001 FALSE FALSE FALSE FALSE     3TC AZT EFV
## 1002 FALSE FALSE FALSE FALSE 3TC AZT RTV SQV
## 1003 FALSE FALSE FALSE FALSE     3TC AZT NVP
## 1004 FALSE FALSE FALSE FALSE     3TC AZT ABC
```

Question 2

14 points

Use the following code to generate data for patients with repeated measures of A1C (a test for levels of blood glucose).

```
genData <- function(n) {
    if(exists(".Random.seed", envir = .GlobalEnv)) {
        save.seed <- get(".Random.seed", envir= .GlobalEnv)
        on.exit(assign(".Random.seed", save.seed, envir = .GlobalEnv))
    } else {
        on.exit(rm(".Random.seed", envir = .GlobalEnv))
    }
    set.seed(n)
    subj <- ceiling(n / 10)
    id <- sample(subj, n, replace=TRUE)
    times <- as.integer(difftime(as.POSIXct("2005-01-01"), as.POSIXct("2000-01-01"), units='secs'))
    dt <- as.POSIXct(sample(times, n), origin='2000-01-01')
    mu <- runif(subj, 4, 10)
    a1c <- unsplit(mapply(rnorm, tabulate(id), mu, SIMPLIFY=FALSE), id)
    data.frame(id, dt, a1c)
}
x <- genData(500)
```

Perform the following manipulations: (2 points each)

2.1. Order the data set by id and dt.

```
x <- x[order(x$id,x$dt),]
```

2.2. For each id, determine if there is more than a one year gap in between observations. Add a new row at the one year mark, with the a1c value set to missing. A two year gap would require two new rows, and so forth.

```
#take subset of observations with a parameter for the id
subset.id <- function(id.num)
{
  return(subset(x,x$id==id.num))
}

#find gaps between observation i and the observation below it that are greater than 365 days (this is w
find.gaps <- function(id)
{
  id.vector <- subset.id(id)
  gaps <- matrix(nrow=nrow(id.vector),ncol=1)
  for (i in 1:nrow(id.vector)-1)
  {
    gaps[i] <- as.integer(difftime(id.vector[i+1,2],id.vector[i,2],units="days")) #days from visit 1 to
  }
  return(gaps)
}

#apply the find.gaps function to all ids 1:50
```

```r
gaps <- NULL
for (id in 1:50)
{
  gaps <- rbind(gaps,find.gaps(id))
}
x <- cbind(x,gaps)

#find rows where the gap (after the observation) > 365
(gaps.positions <- which(abs(x$gaps)>365))
```

```
##  [1]  36  47  49  55  56  69  71  81  87 110 117 119 126 128 135 136 143
## [18] 151 152 159 190 193 196 209 210 215 216 258 267 279 286 294 310 319
## [35] 326 327 336 341 356 360 370 380 381 388 416 420 424 433 436 444 454
## [52] 471 474 481
```

```r
#insert rows for gaps of 1 year
for (i in 1:length(gaps.positions))
{
  row.below <- gaps.positions[i]
  save.above <- x[1:row.below,] #save all of the rows above
  save.below <- x[(row.below+1):nrow(x),] #save all of the rows below
  x[row.below+1,1] <- x[row.below,1]
  x[row.below+1,2] <- x[row.below,2] + days(365)
  x[row.below+1,3] <- NA
  x[row.below+1,4] <- NA
  x <- rbind(save.above,x[row.below+1,],save.below)
  gaps.positions <- gaps.positions + 1 #increment gap indeces to account for new inserted row
}

#apply the function again to allow for two year gaps
gaps2 <- NULL
for (id in 1:50)
{
  gaps2 <- rbind(gaps2,find.gaps(id))
}
x <- cbind(x,gaps2)

(gaps2.positions <- which(abs(x$gaps2)>365)) #find gaps that were greater than 1 year
```

```
## [1] 169 179
```

```r
for (i in 1:length(gaps2.positions))
{
  row.below <- gaps2.positions[i]
  save.above <- x[1:row.below,] #save all of the rows above
  save.below <- x[(row.below+1):nrow(x),] #save all of the rows below
  x[row.below+1,1] <- x[row.below,1]
  x[row.below+1,2] <- x[row.below,2] + days(365)
  x[row.below+1,3] <- NA
  x[row.below+1,4] <- NA
  x <- rbind(save.above,x[row.below+1,],save.below)
  gaps2.positions <- gaps2.positions + 1 #increment gap indeces to account for new inserted row
}
```

2.3. Create a new column visit. For each id, add the visit number. This should be 1 to n where n is the number of observations for an individual. This should include the observations created with missing a1c values.

```
#function that will create a visit field for the parameterized id
count.visits <- function(id)
{
  id.vector <- subset.id(id)
  visit.no <- matrix(nrow=nrow(id.vector),ncol=1)
  for (i in 1:nrow(visit.no))
  {
    visit.no[i] <- i
  }
  return(visit.no)
}

#run the count.visits function on all ids to create a complete column for visits
visit <- NULL
for (id in 1:50)
{
  visit <- rbind(visit,count.visits(id))
}

#add the visit column to the original x dataframe
x <- cbind(x,visit)
```

2.4. For each id, replace missing values with the mean a1c value for that individual.

```
#function that will find NA in the a1c field and replace them with that id's mean a1c
replace.na <- function(id)
{
  id.vector <- subset.id(id)
  a1c.mean <- mean(id.vector$a1c, na.rm=TRUE) #na.rm is important because of the missing values, so if
  new.a1c <- matrix(nrow=nrow(id.vector),ncol=1)
  for (i in 1:nrow(id.vector))
  {
    if (is.na(id.vector[i,3])==TRUE)
    {
      new.a1c[i] <- a1c.mean
    }
    else
    {
      new.a1c[i] <- id.vector[i,3]
    }
  }
  return(new.a1c)
}

#run the replace.na function for every id from 1:50
a1c.replaced <- NULL
for (id in 1:50)
{
  a1c.replaced <- rbind(a1c.replaced,replace.na(id))
}
```

```r
x <- cbind(x,a1c.replaced)
```

2.5. Print mean a1c for each id.

```r
id.means <- function(id)
{
  return(mean(subset.id(id)$a1c.replaced))
}

id <- seq(1:50)

indiv.means <- lapply(id,id.means)

(avg.a1c.by.id <- cbind(id,indiv.means))
```

```
##       id indiv.means
##  [1,] 1  4.063372
##  [2,] 2  7.544643
##  [3,] 3  6.75764
##  [4,] 4  3.892127
##  [5,] 5  9.512311
##  [6,] 6  7.555965
##  [7,] 7  9.161686
##  [8,] 8  7.189064
##  [9,] 9  9.283873
## [10,] 10 7.975217
## [11,] 11 6.917562
## [12,] 12 7.034021
## [13,] 13 9.145282
## [14,] 14 6.623756
## [15,] 15 8.012406
## [16,] 16 4.222158
## [17,] 17 3.996034
## [18,] 18 9.164873
## [19,] 19 5.50721
## [20,] 20 3.726675
## [21,] 21 8.140939
## [22,] 22 5.637501
## [23,] 23 7.366889
## [24,] 24 7.439316
## [25,] 25 6.877135
## [26,] 26 6.556759
## [27,] 27 4.926457
## [28,] 28 7.433917
## [29,] 29 4.508086
## [30,] 30 6.045577
## [31,] 31 7.116586
## [32,] 32 6.568791
## [33,] 33 6.494069
## [34,] 34 6.768615
## [35,] 35 8.4767
```

```
## [36,] 36 9.60441
## [37,] 37 9.606253
## [38,] 38 5.355979
## [39,] 39 6.917013
## [40,] 40 9.530136
## [41,] 41 9.802424
## [42,] 42 3.89177
## [43,] 43 6.095849
## [44,] 44 9.09167
## [45,] 45 6.737204
## [46,] 46 9.621763
## [47,] 47 9.231489
## [48,] 48 6.4046
## [49,] 49 6.096076
## [50,] 50 8.962319
```

2.6. Print total number of visits for each id.

```
total.visits <- function(id)
{
  return(nrow(subset.id(id)))
}
id <- seq(1:50)
total.visits <- lapply(id,total.visits)
(total.visits.by.id <- cbind(id,total.visits))
```

```
##       id total.visits
##  [1,] 1  11
##  [2,] 2  20
##  [3,] 3  14
##  [4,] 4  12
##  [5,] 5  14
##  [6,] 6  10
##  [7,] 7  9
##  [8,] 8  12
##  [9,] 9  11
## [10,] 10 12
## [11,] 11 10
## [12,] 12 10
## [13,] 13 8
## [14,] 14 12
## [15,] 15 8
## [16,] 16 9
## [17,] 17 12
## [18,] 18 10
## [19,] 19 10
## [20,] 20 9
## [21,] 21 10
## [22,] 22 8
## [23,] 23 8
## [24,] 24 15
## [25,] 25 12
## [26,] 26 14
```

```
## [27,] 27 11
## [28,] 28 14
## [29,] 29 10
## [30,] 30 7
## [31,] 31 11
## [32,] 32 5
## [33,] 33 8
## [34,] 34 12
## [35,] 35 11
## [36,] 36 9
## [37,] 37 17
## [38,] 38 15
## [39,] 39 8
## [40,] 40 7
## [41,] 41 17
## [42,] 42 14
## [43,] 43 11
## [44,] 44 11
## [45,] 45 14
## [46,] 46 9
## [47,] 47 12
## [48,] 48 11
## [49,] 49 12
## [50,] 50 10
```

2.7. Print the observations for id = 15.

```r
subset.id(15)
```

```
##       id                  dt      a1c gaps gaps2 visit a1c.replaced
## 11    15 2000-04-30 00:34:50 7.527105  262   262     1     7.527105
## 406   15 2001-01-17 21:11:02 5.898371   97    97     2     5.898371
## 306   15 2001-04-25 06:23:05 8.566593  772   365     3     8.566593
## 484   15 2002-04-25 06:23:05       NA   NA   407     4     8.012406
## 4841  15 2003-04-25 06:23:05       NA   NA   365     5     8.012406
## 48411 15 2003-06-06 14:06:00 9.133769  441   365     6     9.133769
## 263   15 2004-06-05 14:06:00       NA   NA    76     7     8.012406
## 2631  15 2004-08-20 17:47:11 8.936190   NA    NA     8     8.936190
```

Question 3

10 points

Import the addr.txt file from the GitHub repository. This file contains a listing of names and addresses (thanks google). Parse each line to create a data.frame with the following columns: lastname, firstname, streetno, streetname, city, state, zip. Keep middle initials or abbreviated names in the firstname column. Print out the entire data.frame.

```r
addr <- read.delim("https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/addr.txt",
    stringsAsFactors = FALSE, head = FALSE)

removeSpaces <- function(text) {
    for (i in 1:nrow(text)) {
```

```
        text[i, ] <- gsub(" ", "", text[i, ], fixed = TRUE)  #remove blank spaces
    }
    return(text)
}

# find position of capital letters in the space-less string of characters
findCaps <- function(textRow) {
    capsPos <- NULL
    for (i in 1:nchar(textRow)) {
        if (substr(textRow, i, i) %in% LETTERS == TRUE) {
            capsPos <- c(capsPos, i)
        }
    }
    return(capsPos)
}

findPeriods <- function(textRow) {
    periodPos <- NULL
    for (i in 1:nchar(textRow)) {
        if ((substr(textRow, i, i) == ".") == TRUE) {
            periodPos <- c(periodPos, i)
        }
    }
    return(periodPos)
}

findNumbers <- function(textRow) {
    numberPos <- NULL
    for (i in 1:nchar(textRow)) {
        if ((substr(textRow, i, i) %in% seq(0, 9)) == TRUE) {
            numberPos <- c(numberPos, i)
        }
    }
    return(numberPos)
}

get.firstname <- function(textRow) {
    if (length(row.period.positions) > 0) {
        if (row.period.positions[1] - row.cap.positions[3] == 1) {
            firstname[i] <- substr(text[i, ], row.cap.positions[2], row.period.positions[1])  #include
        } else {
            firstname[i] <- substr(text[i, ], row.cap.positions[2], (row.number.positions[1] -
                1))
        }
    } else {
        firstname[i] <- substr(text[i, ], row.cap.positions[2], (row.number.positions[1] -
            1))
    }
}

find.first.jump <- function(rowNums) {
    jump <- vector()
    max.jump <- 0
```

```
    for (i in 1:length(rowNums)) {
        jump[i] <- rowNums[i + 1] - rowNums[i]
    }
    first.jump <- min(rowNums[which(jump > 1)])
    return(first.jump)  #return position of last number in streetno
}

replace.O <- function(textRow) {
    if (substr(addr[41, ], (nchar(addr[41, ]) - 4), (nchar(addr[41, ]) - 4)) ==
        "-") {
        substr(textRow, (nchar(textRow) - 9), nchar(textRow)) <- sub("O", "0",
            substr(textRow, (nchar(textRow) - 9), nchar(textRow)))
    } else {
        substr(textRow, (nchar(textRow) - 4), nchar(textRow)) <- sub("O", "0",
            substr(textRow, (nchar(textRow) - 4), nchar(textRow)))
    }
    return(textRow)
}

lastname <- vector()
firstname <- vector()
streetno <- vector()
streetname <- vector()
city <- vector()
state <- vector()
zip <- vector()

fixText <- function(text) {
    text <- removeSpaces(text)
    for (i in 1:nrow(text)) {
        text[i, ] <- replace.O(text[i, ])  #replace incorrect Os in zipcodes with 0s
        row.cap.positions <- findCaps(text[i, ])
        row.period.positions <- findPeriods(text[i, ])
        row.number.positions <- findNumbers(text[i, ])
        lastname[i] <- substr(text[i, ], 1, (row.cap.positions[2] - 1))
        firstname[i] <- substr(text[i, ], row.cap.positions[2], (row.number.positions[1] -
            1))
        streetno[i] <- substr(text[i, ], row.number.positions[1], find.first.jump(row.number.positions)
        state[i] <- substr(text[i, ], (row.cap.positions[length(row.cap.positions) -
            1]), row.cap.positions[length(row.cap.positions)])
        zip[i] <- substr(text[i, ], (row.cap.positions[length(row.cap.positions)] +
            1), nchar(text[i, ]))
        streetname[i] <- substr(text[i, ], (find.first.jump(row.number.positions) +
            1), (row.cap.positions[length(row.cap.positions) - 2] - 1))
        city[i] <- substr(text[i, ], (row.cap.positions[length(row.cap.positions) -
            2]), (row.cap.positions[length(row.cap.positions) - 1] - 1))
    }
    return(cbind(lastname, firstname, streetno, streetname, city, state, zip))
}

addr <- data.frame(fixText(addr))
```

Question 4

2 points

The first argument to most functions that fit linear models are formulas. The following example defines the response variable death and allows the model to incorporate all other variables as terms. . is used to mean all columns not otherwise in the formula.

```r
url <- "https://github.com/fonnesbeck/Bios6301/raw/master/datasets/haart.csv"
haart_df <- read.csv(url)[,c('death','weight','hemoglobin','cd4baseline')]
coef(summary(glm(death ~ ., data=haart_df, family=binomial(logit))))
```

```
##                 Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept)  3.576411744 1.226870535  2.915069 0.0035561039
## weight      -0.046210552 0.022556001 -2.048703 0.0404911395
## hemoglobin  -0.350642786 0.105064078 -3.337418 0.0008456055
## cd4baseline  0.002092582 0.001811959  1.154872 0.2481427160
```

Now imagine running the above several times, but with a different response and data set each time. Here's a function:

```r
myfun <- function(dat, response) {
  form <- as.formula(response ~ .)
  coef(summary(glm(form, data=dat, family=binomial(logit))))
}
```

Unfortunately, it doesn't work. tryCatch is "catching" the error so that this file can be knit to PDF.

```r
tryCatch(myfun(haart_df, death), error = function(e) e)
```

```
## <simpleError in eval(expr, envir, enclos): object 'death' not found>
```

What do you think is going on? Consider using debug to trace the problem.

The tryCatch error message reads: <simpleError in eval(expr, envir, enclos): object 'death' not found>, so I wondered if the problem could be in the way that the "response" variable was included in the parameters. To test my theory, I ran the line again with a minor correction:

```r
tryCatch(myfun(haart_df, haart_df$death), error = function(e) e)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
##                  Estimate  Std. Error       z value  Pr(>|z|)
## (Intercept) -2.656607e+01 115935.1724 -2.291459e-04 0.9998172
## death        5.313214e+01  69028.4183  7.697140e-04 0.9993859
## weight      -4.499694e-15   1939.0571 -2.320558e-18 1.0000000
## hemoglobin   5.124642e-14   9774.8190  5.242697e-18 1.0000000
## cd4baseline  1.830771e-16    184.0846  9.945271e-19 1.0000000
```

Hooray! Now the function runs completely. Although, the algorithm does not converge, but that is more an artifact of the data than the code written above.

Bonus

5 bonus points

Create a working function.

```
calculate.tax <- function(price)
{
  return(price*1.0925) #returns the price after accounting for Nashville's absurd 9.25% sales tax
}
starbucks <- 4 #the price of my morning vanilla sweet cream cold brew (before tax)
calculate.tax(starbucks) #price of my morning vanilla sweet cream cold brew (after tax)
```

```
## [1] 4.37
```