Homework_2

Sam Budoff

10/16/2014

Homework 2

Sam Budoff, 10.16.14

Question 1

A problem with the Newton-Raphson algorithm is that it needs the derivative f'. If the derivative is hard to compute or does not exist, then we can use the *secant method*, which only requires that the function f is continuous.

Like the Newton-Raphson method, the **secant method** is based on a linear approximation to the function f. Suppose that f has a root at a. For this method we assume that we have *two* current guesses, x_0 and x_1, for the value of a. We will think of x_0 as an older guess and we want to replace the pair x_0, x_1 by the pair x_1, x_2, where x_2 is a new guess.

To find a good new guess x2 we first draw the straight line from $(x_0, f(x_0))$ to $(x_1, f(x_1))$, which is called a secant of the curve $y = f(x)$. Like the tangent, the secant is a linear approximation of the behavior of $y = f(x)$, in the region of the points x_0 and x_1. As the new guess we will use the x-coordinate x_2 of the point at which the secant crosses the x-axis.

The general form of the recurrence equation for the secant method is:

$$x_{i+1} = x_i - f(x_i)\frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})}$$

Notice that we no longer need to know f' but in return we have to provide *two* initial points, x_0 and x_1.

Write a function that implements the secant algorithm. Validate your program by finding the root of the function $f(x) = \cos(x) - x$.

```
f<-function(x){(cos(x)-x)}
sec<-function(i,l) {
  repeat {
    if(abs(f(l))>10^-7) {
      k=i
      i=l - (f(l)*((l - i)/(f(l) - f(i))))
      l=k
    } else {break}
  }
  return(i)
}
```

Running this code with the given function and arbitrarily chosen x_0 and x_1 equal to -20 and +20, respectively, reveals the root:

```
sec(-20,20)
```

```
## [1] 0.7391
```

Compare its performance with the Newton-Raphson method – which is faster, and by how much? For this example $f'(x) = -\sin(x) - 1$. **Using the Newton-Raphson method, based on the code provided by Dr. Beck but using the given function, reveals the root:**

```
f=function(x) {-sin(x) - 1}

NR<- function(i) {
  while(abs(-sin(i)-1) > 10^-7) {
    f <- -sin(i)-1
    fp <- -cos(i)
    i <- i - f/fp
 }
i
}
```

```
x=1
lx=0
  print('Newton-Raphson')
```

```
## [1] "Newton-Raphson"
```

```
  NR(x)
```

```
## [1] -1.571
```

```
  NRt<-system.time(replicate(10000,NR(x)))
  print('Secant')
```

```
## [1] "Secant"
```

```
  sec(x,lx)
```

```
## [1] -1.571
```

```
  SECt<-system.time(replicate(10000,sec(x,lx)))
  print('Secant, Newton-Raphson time difference on 10000 trials')
```

```
## [1] "Secant, Newton-Raphson time difference on 10000 trials"
```

```
  SECt[3]-NRt[3]
```

```
## elapsed
##   0.891
```

Based on these results it appears that the standard Newton-Raphson Method is less efficient then my function by 0.959 time units.

Question 2

Import the HAART dataset (`haart.csv`) from the GitHub repository into R, and perform the following manipulations: **For the following questions, all code and answers are output by the function(s) following question 6, but a print output will delineate between results based on the question they pertain to.**

1. Convert date columns into a usable (for analysis) format.
2. Create an indicator variable (one which takes the values 0 or 1 only) to represent death within 1 year of the initial visit. **The columb "t2death" in the dataframe "dates2.df" displays a 1 for individuals who died within one year of their initial visit.**
3. Use the `init.date`, `last visit` and `death.date` to calculate a followup time, which is the difference between the first and either the last visit or a death event (whichever comes first). If these times are longer than 1 year, censor them.
4. Create another indicator variable representing loss to followup; that is, if their status 1 year after the first visit was unknown. **Status of 1 is indicative of lack of followup whcih is assumed to mean unknown disease progression. 0 indicates known followup, or known death**
5. 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.
6. 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!).

```
PS2<-function(file,drugz) {

haart.csv<-read.csv(file)

haart.df<-data.frame(haart.csv)
init.date<-haart.df$init.date
last.visit<-haart.df$last.visit
date.death<-haart.df$date.death
dates.df<-data.frame(init.date,last.visit,date.death)

print("Part 1")


dates.df$date.death <- as.POSIXlt(dates.df$date.death, format="%m/%d/%y")
dates.df$last.visit <- as.POSIXlt(dates.df$last.visit, format="%m/%d/%y")
dates.df$init.date <- as.POSIXlt(dates.df$init.date, format="%m/%d/%y")

t2death<-difftime(dates.df$date.death, dates.df$init.date, units="weeks")

t2death[is.na(t2death)]<-100
t2death<-ifelse(t2death<=52,1,0)

dates2.df<-data.frame(init.date,last.visit,date.death, t2death)

print(dates2.df$t2death)

print("Part 2")


dates.df$last.visit <- as.POSIXct(dates.df$last.visit, format="%m/%d/%y")
```

```
dates.df$init.date <- as.POSIXct(dates.df$init.date, format="%m/%d/%y")
dates.df$date.death <- as.POSIXct(dates.df$date.death, format="%m/%d/%y")

checkup<-ifelse(haart.df$death==0,(dates.df$last.visit-dates.df$init.date),0)

followup<-ifelse(checkup<=3.15569e7,1,0)

haartI.df<-data.frame(init.date,last.visit,date.death, t2death, followup)

print(haartI.df$followup)

print("Part 3")


lost<-c(rep(0,length((init.date))))
lost<-ifelse(haart.df$death==1,lost+2,lost+0)
#2 represents dead
lost<-ifelse((dates.df$init.date)==(dates.df$last.visit),lost+1,lost+0)
lost<-ifelse(lost==1,1,0)
lost[is.na(lost)]<-0


haartI.df<-data.frame(init.date,last.visit,date.death, t2death, followup, lost)

print(haartI.df$lost)

print("Part 4")


init.reg <- as.character(haart.df$init.reg)
haartI.df[['init.reg_list2']] <- strsplit(init.reg, ",")
all_drugs <- unique(unlist(haartI.df$init.reg))
all_drugs<-ifelse(all_drugs<drugz,drugz,all_drugs)
reg_drugs <- matrix(FALSE, nrow=nrow(haartI.df), ncol=length(all_drugs))
for(i in seq_along(all_drugs)) {
  reg_drugs[,i] <- sapply(haartI.df$init.reg_list2, function(x) all_drugs[i] %in% x)
}

reg_drugs <- data.frame(reg_drugs)
names(reg_drugs) <- all_drugs
haartI.df <- cbind(haartI.df, reg_drugs)

print(colnames(haartI.df))

print("Part 5")
return(haartI.df)
}

haart.csv<-read.csv('haart.csv')
h2.csv<-read.csv('haart2.csv')

drugz=0
H1<-PS2('haart.csv', drugz)
```

```
## [1] "Part 1"
##    [1] 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0
##   [35] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0
##   [69] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [103] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [137] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
##  [171] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [205] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [239] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [273] 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
##  [307] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 0 0 1 0 1 0
##  [341] 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0
##  [375] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [409] 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
##  [443] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1
##  [477] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [511] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
##  [545] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0
##  [579] 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
##  [613] 1 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [647] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
##  [681] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0
##  [715] 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [749] 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0
##  [783] 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [817] 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [851] 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0
##  [885] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [919] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0
##  [953] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1
##  [987] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
## [1] "Part 2"
##    [1] 0 0 1 1 0 0 0 0 0 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 0 0
##   [35] 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0
##   [69] 0 1 0 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0
##  [103] 1 1 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0
##  [137] 0 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 1 1 0 0
##  [171] 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 1 0 1 1 0 0 1 0 0 0 0 0
##  [205] 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0
##  [239] 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0
##  [273] 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0
##  [307] 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 1 0 1 1
##  [341] 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1
##  [375] 1 0 0 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0
##  [409] 1 1 0 0 0 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0 0 0
##  [443] 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 1
##  [477] 0 1 1 1 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0
##  [511] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 1 0 0 0
##  [545] 0 0 0 1 0 0 0 0 1 0 1 0 1 0 0 0 0 1 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 0
##  [579] 1 0 1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 1 1 0 0 0 0 0 0 0 1 0
##  [613] 1 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0
##  [647] 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0
##  [681] 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0
##  [715] 0 1 0 0 0 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0
```

```
##  [749] 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0
##  [783] 1 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 0 0 0 0 1 0 1 0
##  [817] 1 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 1 1
##  [851] 0 0 1 1 0 1 0 1 0 1 1 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0
##  [885] 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 0 0 0 1
##  [919] 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 0
##  [953] 0 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1
##  [987] 0 1 0 0 1 0 0 0 0 0 0 1 0 0 0
## [1] "Part 3"
##    [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [35] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##   [69] 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
##  [103] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
##  [137] 0 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [171] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0
##  [205] 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [239] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0
##  [273] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [307] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [341] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1
##  [375] 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [409] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [443] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [477] 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
##  [511] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [545] 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0
##  [579] 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
##  [613] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [647] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0
##  [681] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [715] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [749] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [783] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [817] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [851] 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [885] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1
##  [919] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [953] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
##  [987] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [1] "Part 4"
##  [1] "init.date"      "last.visit"     "date.death"     "t2death"
##  [5] "followup"       "lost"           "init.reg_list2" "3TC"
##  [9] "AZT"            "EFV"            "NVP"            "D4T"
## [13] "ABC"            "DDI"            "IDV"            "LPV"
## [17] "RTV"            "SQV"            "FTC"            "TDF"
## [21] "DDC"            "NFV"            "T20"            "ATV"
## [25] "FPV"
## [1] "Part 5"
```

```
unique_drugs<-function(haartI.df){
  all_drugs <- unique(unlist(haartI.df$init.reg))
  return(all_drugs)
}
```

```
drugz<-unique_drugs(H1)
H2<-PS2('haart2.csv',drugz)
```

```
## [1] "Part 1"
## [1] 0 0 0 0
## [1] "Part 2"
## [1] 1 1 1 1
## [1] "Part 3"
## [1] 0 0 0 0
## [1] "Part 4"
##  [1] "init.date"      "last.visit"     "date.death"     "t2death"
##  [5] "followup"       "lost"           "init.reg_list2" "3TC"
##  [9] "AZT"            "NVP"            "NVP"            "EFV"
## [13] "D4T"            "DDI"            "IDV"            "NVP"
## [17] "RTV"            "SQV"            "FTC"            "TDF"
## [21] "DDC"            "NVP"            "T20"            "EFV"
## [25] "FPV"
## [1] "Part 5"
```

```
names(H1) <- names(H2)
#identical(names(H1), names(H2))

haart2 <- rbind(H1,H2)
nrow(haart2)
```

```
## [1] 1004
```

```
ncol(haart2)
```

```
## [1] 25
```

```
print("Problem Set 2 Section 2 Complete")
```

```
## [1] "Problem Set 2 Section 2 Complete"
```

Question 3 ###3

12 points

Obtain a copy of the football-values lecture. Save the five CSV files in your working directory.

Modify the code to create a function. This function will create dollar values given information (as arguments) about a league setup. It will return a data.frame and write this data.frame to a CSV file. The final data.frame should contain the columns 'PlayerName', 'pos', 'points', 'value' and be orderd by value descendingly. Do not round dollar values.

Note that the returned data.frame should have `sum(posReq)*nTeams` rows.

Define the function as such (6 points):

```
# path: directory path to input files
# file: name of the output file; it should be written to path
# nTeams: number of teams in league
# cap: money available to each team
# posReq: number of starters for each position
# points: point allocation for each category
ffvalues <- function(path, file='outfile.csv', nTeams=12, cap=200, posReq=c(qb=1, rb=2, wr=3, te=1, k=1
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                              rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)) {
  ## read in CSV files
  ## calculate dollar values
  ## save dollar values as CSV file
  ## return data.frame with dollar values
}
```

```
path="/home/sam/Documents/Bios301/football/"
ffvalues <- function(path, file='outfile.csv', nTeams=12, cap=200, posReq=c(qb=1, rb=2, wr=3, te=1, k=1
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                              rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)) {
  setwd(file.path(path))

  k <- read.csv('proj_k14.csv', header=TRUE, stringsAsFactors=FALSE)
  qb <- read.csv('proj_qb14.csv', header=TRUE, stringsAsFactors=FALSE)
  rb <- read.csv('proj_rb14.csv', header=TRUE, stringsAsFactors=FALSE)
  te <- read.csv('proj_te14.csv', header=TRUE, stringsAsFactors=FALSE)
  wr <- read.csv('proj_wr14.csv', header=TRUE, stringsAsFactors=FALSE)

  cols <- unique(c(names(k), names(qb), names(rb), names(te), names(wr)))

  k[,'pos'] <- 'k'
  qb[,'pos'] <- 'qb'
  rb[,'pos'] <- 'rb'
  te[,'pos'] <- 'te'
  wr[,'pos'] <- 'wr'

  cols <- c(cols, 'pos')

k[,setdiff(cols, names(k))] <- 0
qb[,setdiff(cols, names(qb))] <- 0
rb[,setdiff(cols, names(rb))] <- 0
te[,setdiff(cols, names(te))] <- 0
wr[,setdiff(cols, names(wr))] <- 0

x <- rbind(k[,cols], qb[,cols], rb[,cols], te[,cols], wr[,cols])


sub('.', '', names(x))

names(x) <- gsub('[.]', '', names(x))

x[,'p_fg'] <- x[,'fg']*4
x[,'p_xpt'] <- x[,'xpt']*1
x[,'p_pass_yds'] <- x[,'pass_yds']/25
```

```
x[,'p_pass_tds'] <- x[,'pass_tds']*4

x[,'p_pass_ints'] <- x[,'pass_ints']*-2
x[,'p_rush_yds'] <- x[,'rush_yds']/10
x[,'p_rush_tds'] <- x[,'rush_tds']*6
x[,'p_fumbles'] <- x[,'fumbles']*-2
x[,'p_rec_yds'] <- x[,'rec_yds']/20
x[,'p_rec_tds'] <- x[,'rec_tds']*6

x[,'points'] <- rowSums(x[,grep("^p_", names(x))])

x2 <- x[order(x[,'points'], decreasing=TRUE),]
outfile.csv<-x2

return('outfile.csv')
}
```

1. Call `x1 <- ffvalues('.')`
    1. How many players are worth more than $20? (1 point)
    2. Who is 15th most valuable running back (rb)? (1 point)
2. Call `x2 <- ffvalues(getwd(), '16team.csv', nTeams=16, cap=150)`
    1. How many players are worth more than $20? (1 point)
    2. How many wide receivers (wr) are in the top 40? (1 point)
3. Call:

```
x3 <- ffvalues('.', 'qbheavy.csv', posReq=c(qb=2, rb=2, wr=3, te=1, k=0),
        points=c(fg=0, xpt=0, pass_yds=1/25, pass_tds=6, pass_ints=-2,
                rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6))
```

    1. How many players are worth more than $20? (1 point)
    2. How many quarterbacks (qb) are in the top 30? (1 point)