Bios 6301: Assignment 5

Nick Strayer

Question 1

24 points

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

```
setwd("/Users/Nick/Dropbox/vandy/computing/Bios6301/datasets")
h <- read.csv("haart.csv", stringsAsFactors = F)
library(lubridate)
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.1.3
```

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

```
fix_dates = function(h){
  fix1900s <- function(x, year=16){ #Make it 1900s if the 10s digits are above 16.
    m <- year(x) %% 100
    year(x) <- ifelse(m > year, 1900+m, 2000+m)
    x
  }
  h[ , "init.date"]  <- fix1900s(mdy(h[, "init.date"]))
  h[ , "last.visit"] <- fix1900s(mdy(h[, "last.visit"]))
  h[ , "date.death"] <- fix1900s(mdy(h[, "date.death"]))
  h
}
h = fix_dates(h)

table(format(h[ , "init.date"], "%Y"))
```

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

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?

We check to see if the `date.death` column is a real date, then if it is we assign `1` if they died within a year and `0` if they either died more than a year later or there was no death reported.

```
death_in_year = function(h){
  h[,"death.in.year"] <- ifelse(!is.na(h[ , "date.death"]), + h[ , "date.death"] - h[ , "init.date"] < 3
  h
}
h = death_in_year(h)

sum(h[,"death.in.year"])
```

```
## [1] 92
```

So we see that there were 92 deaths within a year.

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). If these times are longer than 1 year, censor them (this means if the value is above 365, set followup to 365). Print the quantile for this new variable.*

```
followup_time = function(h){
  for(i in 1:dim(h)[1]){
  dif <- NULL #initialize difference
  #if last visit is not na calc the difference
  if(!is.na(h[i, "last.visit"])) dif <- difftime(h[i, "last.visit"], h[i,"init.date"], units = "days")
  #if the date death is not na make the difference the minimum between the old dif and the new one.
  if(!is.na(h[i, "date.death"])) dif <- min(dif, difftime(h[i, "date.death"], h[i,"init.date"], units =

  h[i,"followup.time"] <- min(365, dif)
  }
  h
}
h = followup_time(h)

quantile(h[,"followup.time"])
```

```
##     0%    25%    50%    75%   100%
##   0.00 320.75 365.00 365.00 365.00
```

4. 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?

```
loss_to_followup = function(h){
  h[,"loss.to.followup"] <- ifelse(is.na(h[,"date.death"]) & h[,"followup.time"] < 365, 1,0)
  h
}
h = loss_to_followup(h)
```

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. Which drug regimen are found over 100 times?*

```
drug_counts = function(h){
  #grab unique drug names
  drugs <- unique(unlist(sapply(h[,"init.reg"], function(d) unlist(strsplit(d, ",") ) ))) #this is ugly

  for(drug in drugs) h[,drug] = 0 #add empty columns

  for(i in 1:dim(h)[1]){ #for each row in the dataframe
    for(drug in drugs){
      if(drug %in% unlist(strsplit(h[i,"init.reg"], ",") )) h[i, drug] = 1 #if the drug is there add to
    }
  }
  for(drug in drugs) if(sum(h[, drug]) > 100) print(drug) #Print the drugs that are prescribed more tha
```

```
  h
}
h = drug_counts(h)
```

```
## [1] "3TC"
## [1] "AZT"
## [1] "EFV"
## [1] "NVP"
## [1] "D4T"
```

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!). Show the first five records and the last five records of the complete (and clean) data set.*

```
setwd("/Users/Nick/Dropbox/vandy/computing/Bios6301/datasets")
h0 <- read.csv("haart.csv", stringsAsFactors = F) #Read in the two datasets
h1 <- read.csv("haart2.csv", stringsAsFactors = F)
h2  <- rbind(h0,h1) #Merge them

h2 = fix_dates(h2) #Run all the previously written functions on the new data.
h2 = death_in_year(h2)
h2 = followup_time(h2)
h2 = loss_to_followup(h2)
h2 = drug_counts(h2)
```

```
## [1] "3TC"
## [1] "AZT"
## [1] "EFV"
## [1] "NVP"
## [1] "D4T"
```

```
kable(h2[1:5,])
```

male	age	aids	cd4baseline	logvl	weight	hemoglobin	init.reg	init.date	last.visit	death	date
1	25	0	NA	NA	NA	NA	3TC,AZT,EFV	2003-07-01	2007-02-26	0	NA
1	49	0	143	NA	58.0608	11	3TC,AZT,EFV	2004-11-23	2008-02-22	0	NA
1	42	1	102	NA	48.0816	1	3TC,AZT,EFV	2003-04-30	2005-11-21	1	2006
0	33	0	107	NA	46.0000	NA	3TC,AZT,NVP	2006-03-25	2006-05-05	1	2006
1	27	0	52	4	NA	NA	3TC,D4T,EFV	2004-09-01	2007-11-13	0	NA

```
rows = dim(h2)[1]
kable(h2[(rows-5):rows,])
```

	male	age	aids	cd4baseline	logvl	weight	hemoglobin	init.reg	init.date	last.visit
999	0	31.00000	0	102	NA	61.6896	11	3TC,AZT,NVP	2003-05-22	2008-03-0
1000	0	40.00000	1	131	NA	46.2672	8	3TC,D4T,NVP	2003-07-03	2008-02-2
1001	0	27.00000	0	232	NA	NA	NA	3TC,AZT,NVP	2003-12-01	2004-01-0

	male	age	aids	cd4baseline	logvl	weight	hemoglobin	init.reg	init.date	last.visit
1002	1	38.72142	0	170	NA	84.0000	NA	3TC,AZT,NVP	2002-09-26	2004-03-2
1003	1	23.00000	NA	154	3.995635	65.5000	14	3TC,DDI,EFV	2007-01-31	2007-04-1
1004	0	31.00000	0	236	NA	45.8136	NA	3TC,D4T,NVP	2003-12-03	2007-10-1

Question 2

10 points

Obtain the code for using Newton's Method to estimate logistic regression parameters (`logistic.r`) and modify it to predict `death` from `weight`, `hemoglobin` and `cd4baseline` in the HAART dataset. Use complete cases only. Report the estimates for each parameter, including the intercept.

```
toRegress = h[,c("death", "weight", "hemoglobin", "cd4baseline")] #grab the data.

toRegress = toRegress[complete.cases(toRegress),] #Get rid of NAs

x <- toRegress[2:4]
y <- toRegress[1]

estimate_logistic <- function(x, y, MAX_ITER=10) {

    logistic <- function(x) 1 / (1 + exp(-x))

    n <- dim(x)[1]
    k <- dim(x)[2]

    x <- as.matrix(cbind(rep(1, n), x))
    y <- as.matrix(y)

    # Initialize fitting parameters
    theta <- rep(0, k+1)

    J <- rep(0, MAX_ITER)

    for (i in 1:MAX_ITER) {

        # Calculate linear predictor
        z <- x %*% theta

        # Apply logit function
        h <- logistic(z)

        # Calculate gradient
        grad <- t((1/n)*x) %*% as.matrix(h - y)

        # Calculate Hessian
        H <- t((1/n)*x) %*% diag(array(h)) %*% diag(array(1-h)) %*% x

        # Calculate log likelihood
        J[i] <- (1/n) %*% sum(-y * log(h) - (1-y) * log(1-h))

        # Newton's method
```

```r
        theta <- theta - solve(H) %*% grad
    }
    return(theta)
}

estimate_logistic(x, y)
```

```
##                    [,1]
## rep(1, n)    3.576411744
## weight      -0.046210552
## hemoglobin  -0.350642786
## cd4baseline  0.002092582
```

Question 3

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
setwd("/Users/Nick/Dropbox/vandy/computing/Bios6301/datasets")
addr <- readLines("addr.txt") #read it in.
res  <- lapply(addr, function(s){unlist(strsplit(s, split = "[ ]{2,}"))})
df   <- do.call(rbind.data.frame, res)
names(df) <- c("lastname", "firstname", "address", "city", "state", "zip")
df[] <- lapply(df, as.character) #strings as factors = always and forever false.

#Now we have to fix the first name column and split the adress one.

#first names:
df$firstname  <- sapply(df$firstname, function(d) return(strsplit(d, " ")[[1]][1])) #Get rid of middle
df$streetno   <- sapply(df$address, function(d) return(strsplit(d, " ")[[1]][1])) #grab street number
df$streetname <- gsub("[0-9]{1,} ", "", df$address) #grab street name
df$address    <- NULL #get rid of address variable.
kable(df)
```

lastname	firstname	city	state	zip	streetno	streetname
Bania	Thomas	Boston	MA	O2215	725	Commonwealth Ave.
Barnaby	David	Wms. Bay	WI	53191	373	W. Geneva St.
Bausch	Judy	Wms. Bay	WI	53191	373	W. Geneva St.
Bolatto	Alberto	Boston	MA	O2215	725	Commonwealth Ave.
Carlstrom	John	Chicago	IL	60637	933	E. 56th St.
Chamberlin	Richard	Hilo	HI	96720	111	Nowelo St.
Chuss	Dave	Evanston	IL	60208-3112	2145	Sheridan Rd
Davis	E.	Chicago	IL	60637	933	E. 56th St.
Depoy	Darren	Columbus	OH	43210	174	W. 18th Ave.
Griffin	Greg	Pittsburgh	PA	15213	5000	Forbes Ave.
Halvorsen	Nils	Chicago	IL	60637	933	E. 56th St.
Harper	Al	Wms. Bay	WI	53191	373	W. Geneva St.
Huang	Maohai	Boston	MA	O2215	725	W. Commonwealth Ave.
Ingalls	James	Boston	MA	O2215	725	W. Commonwealth Ave.

lastname	firstname	city	state	zip	streetno	streetname
Jackson	James	Boston	MA	O2215	725	W. Commonwealth Ave.
Knudsen	Scott	Wms. Bay	WI	53191	373	W. Geneva St.
Kovac	John	Chicago	IL	60637	5640	S. Ellis Ave.
Landsberg	Randy	Chicago	IL	60637	5640	S. Ellis Ave.
Lo	Kwok-Yung	Urbana	IL	61801	1002	W. Green St.
Loewenstein	Robert	Wms. Bay	WI	53191	373	W. Geneva St.
Lynch	John	Arlington	VA	22230	4201	Wilson Blvd
Martini	Paul	Columbus	OH	43210	174	W. 18th Ave.
Meyer	Stephan	Chicago	IL	60637	933	E. 56th St.
Mrozek	Fred	Wms. Bay	WI	53191	373	W. Geneva St.
Newcomb	Matt	Pittsburgh	PA	15213	5000	Forbes Ave.
Novak	Giles	Evanston	IL	60208-3112	2145	Sheridan Rd
Odalen	Nancy	Wms. Bay	WI	53191	373	W. Geneva St.
Pernic	Dave	Wms. Bay	WI	53191	373	W. Geneva St.
Pernic	Bob	Wms. Bay	WI	53191	373	W. Geneva St.
Peterson	Jeffrey	Pittsburgh	PA	15213	5000	Forbes Ave.
Pryke	Clem	Chicago	IL	60637	933	E. 56th St.
Rebull	Luisa	Chicago	IL	60637	5640	S. Ellis Ave.
Renbarger	Thomas	Evanston	IL	60208-3112	2145	Sheridan Rd
Rottman	Joe	Littleton	CO	80125	8730	W. Mountain View Ln
Schartman	Ethan	Chicago	IL	60637	933	E. 56th St.
Spotz	Bob	Wms. Bay	WI	53191	373	W. Geneva St.
Thoma	Mark	Wms. Bay	WI	53191	373	W. Geneva St.
Walker	Chris	Tucson	AZ	85721	933	N. Cherry St.
Wehrer	Cheryl	Pittsburgh	PA	15213	5000	Forbes Ave.
Wirth	Jesse	Wms. Bay	WI	53191	373	W. Geneva St.
Wright	Greg	Holmdel	NY	O7733-1988	791	Holmdel-Keyport Rd.
Zingale	Michael	Chicago	IL	60637	5640	S. Ellis Ave.

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.

```
# 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))))
```

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

```
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 is.data.frame(data): object 'haart_df' not found>
```

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

5 bonus points

Create a working function.