Bios 6301: Assignment 5

Josh DeClercq

Grade 55/50

Great job, really well done. Check out how Cole approache Question 2 using lapply and tapply.

Note: In the future, for packages that might not be installed by collaborators, you can use the following to check for and install a package:

```
if("lubridate" %in% rownames(installed.packages()) == FALSE) {
  install.packages("lubridate",repos="http://cran.rstudio.com/")
}
```

Due Tuesday, 15 November, 1:00 PM

$5^{n=day}$ points taken off for each day late.

50 points total.

Submit a single knitr file (named `homework5.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 `homework5.rmd` or include author name may result in 5 points taken off.

Question 1

24 points

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

```
file <- "https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/haart.csv"
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:base':
##
##     date
```

```
haart1Raw <- read.csv(file, sep = ",", stringsAsFactors = FALSE)

cleanData <- function(haartdata){
  #Question 1.1
  haartdata[,'last.visit'] <- as.Date(haartdata[,'last.visit'], format = '%m/%d/%y')
  haartdata[,'init.date'] <- as.Date(haartdata[,'init.date'],  format = '%m/%d/%y')
  haartdata[,'date.death'] <- as.Date(haartdata[,'date.death'],  format = '%m/%d/%y')

  #Question 1.2
  haartdata[,'one.year.death']<-
    as.numeric(difftime(haartdata[,'date.death'],haartdata[,'init.date'],units="days") < 365)
  haartdata[,'one.year.death'][which(is.na(haartdata[,'one.year.death']))] <- 0
```

```
  #Question 1.3
  haartdata[,'lower'] <- pmin(haartdata[,'last.visit'],haartdata[,'date.death'], na.rm = TRUE)
  haartdata[,'follow.time'] <- difftime(haartdata[,'lower'],
                                        haartdata[,'init.date'], units = 'days' )
  haartdata[,'follow.time'][which(haartdata[,'follow.time'] > 365)] <- 365

  #Question 1.4
  haartdata[,'lost.follow'] <- 0
  haartdata[,'lost.follow'][which(haartdata[,'follow.time'] < 365 &
                                    haartdata[,'death'] == 0)] <- 1

  #Question 1.5
  init.reg <- as.character(haartdata[,'init.reg'])
  all.reg <- unique(unlist(strsplit(init.reg, ",")))
  row.reg <- strsplit(init.reg, ",")

  dim(sapply(all.reg, function(j) sapply(row.reg, function(i) j %in% i)))
  user.reg <- sapply(all.reg, function(j) sapply(row.reg, function(i) j %in% i))
  haartdata<- cbind(haartdata, user.reg)

  return(haartdata)
}
haart1 <- cleanData(haart1Raw)
```

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

```
table(year(haart1[,'init.date']))
```

```
##
## 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?

```
sum(haart1[,'one.year.death'])
```

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

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.

```
quantile(haart1[,'follow.time'])
```

```
## Time differences in days
##     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?

```
sum(haart1[,'lost.follow'])
```

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

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?

```
threeTC <- which(colnames(haart1) == '3TC')
FPV <- which(colnames(haart1) == 'FPV')
names(which(colSums(haart1[,c(threeTC:FPV)])>100))
```

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

```
file2 <- 'https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/haart2.csv'
haart2Raw <- read.csv(file2, sep = ",", stringsAsFactors = FALSE)
combinedRaw <- rbind(haart1Raw, haart2Raw)
# haart2

file2 <- 'https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/haart2.csv'
combinedClean <- cleanData(combinedRaw)

combinedClean[c(1:5, (nrow(combinedClean)-4):nrow(combinedClean) ),]
```

```
##      male      age aids cd4baseline    logvl  weight hemoglobin
## 1       1 25.00000    0          NA       NA      NA         NA
## 2       1 49.00000    0         143       NA 58.0608         11
## 3       1 42.00000    1         102       NA 48.0816          1
## 4       0 33.00000    0         107       NA 46.0000         NA
## 5       1 27.00000    0          52 4.000000      NA         NA
## 1000    0 40.00000    1         131       NA 46.2672          8
## 1001    0 27.00000    0         232       NA      NA         NA
## 1002    1 38.72142    0         170       NA 84.0000         NA
## 1003    1 23.00000   NA         154 3.995635 65.5000         14
## 1004    0 31.00000    0         236       NA 45.8136         NA
##         init.reg  init.date last.visit death date.death one.year.death
## 1    3TC,AZT,EFV 2003-07-01 2007-02-26     0       <NA>              0
## 2    3TC,AZT,EFV 2004-11-23 2008-02-22     0       <NA>              0
## 3    3TC,AZT,EFV 2003-04-30 2005-11-21     1 2006-01-11              0
## 4    3TC,AZT,NVP 2006-03-25 2006-05-05     1 2006-05-07              1
## 5    3TC,D4T,EFV 2004-09-01 2007-11-13     0       <NA>              0
## 1000 3TC,D4T,NVP 2003-07-03 2008-02-29     0       <NA>              0
## 1001 3TC,AZT,NVP 2003-12-01 2004-01-05     0       <NA>              0
## 1002 3TC,AZT,NVP 2002-09-26 2004-03-29     0       <NA>              0
## 1003 3TC,DDI,EFV 2007-01-31 2007-04-16     0       <NA>              0
## 1004 3TC,D4T,NVP 2003-12-03 2007-10-11     0       <NA>              0
##           lower follow.time lost.follow  3TC   AZT   EFV   NVP   D4T   ABC
## 1    2007-02-26    365 days           0 TRUE  TRUE  TRUE FALSE FALSE FALSE
## 2    2008-02-22    365 days           0 TRUE  TRUE  TRUE FALSE FALSE FALSE
## 3    2005-11-21    365 days           0 TRUE  TRUE  TRUE FALSE FALSE FALSE
## 4    2006-05-05     41 days           0 TRUE  TRUE FALSE  TRUE FALSE FALSE
## 5    2007-11-13    365 days           0 TRUE FALSE  TRUE FALSE  TRUE FALSE
## 1000 2008-02-29    365 days           0 TRUE FALSE FALSE  TRUE  TRUE FALSE
## 1001 2004-01-05     35 days           1 TRUE  TRUE FALSE  TRUE FALSE FALSE
## 1002 2004-03-29    365 days           0 TRUE  TRUE FALSE  TRUE FALSE FALSE
```

```
## 1003 2007-04-16     75 days           1 TRUE FALSE  TRUE FALSE FALSE FALSE
## 1004 2007-10-11    365 days           0 TRUE FALSE FALSE  TRUE  TRUE FALSE
##        DDI   IDV   LPV   RTV   SQV   FTC   TDF   DDC   NFV   T20   ATV
## 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
## 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 FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1003  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## 1004 FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##        FPV
## 1    FALSE
## 2    FALSE
## 3    FALSE
## 4    FALSE
## 5    FALSE
## 1000 FALSE
## 1001 FALSE
## 1002 FALSE
## 1003 FALSE
## 1004 FALSE
```

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)

1. Order the data set by `id` and `dt`.

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

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.

```
di <- numeric(500)

for (i in 2:500){
  t1 <- xsort[i,'dt']
  t2 <- xsort[i-1,'dt']
  if (xsort[i-1,'id'] == xsort[i,'id']){
  di[i] <- difftime(t1,t2  , units = 'days')
  }
}

for( i in length(di):1){
  if (floor(di/365)[i]>0){
    for(j in floor(di/365)[i]:1){
    newRow <- list(xsort[,'id'][i-1],xsort[,'dt'][i-1] + years(j), NA)
    xsort <- rbind(xsort[1:i-1,], newRow, xsort[i:length(xsort[,'id']),])
    }
  }
}
```

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.

```
xsort[,'visit'] <- 1
for (i in 2:nrow(xsort)){
  if (xsort[,'id'][i-1] == xsort[,'id'][i]){
    xsort[,'visit'][i] <- xsort[,'visit'][i-1] + 1
  }
}
```

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

```
A1C <- 0
for (i in 1:nrow(xsort)){
    A1C[i] <- mean(xsort[,'a1c'][xsort[,'id'] == xsort[,'id'][i]], na.rm = TRUE)
}
xsort[,'a1c'][which(is.na(xsort[,'a1c']))] <-  A1C[which(is.na(xsort[,'a1c']))]
```

5. Print mean `a1c` for each `id`.

```
a1cMean <- 0
for (i in 1:50){
  a1cMean[i] <- mean(as.numeric(xsort[,'a1c'][xsort[,'id'] == i]), na.rm= TRUE)
}
a1cdf <- cbind(unique(xsort[,'id']),a1cMean)
colnames(a1cdf) <- c('id', 'mean a1c')
a1cdf
```

```
##       id mean a1c
##  [1,]  1 4.063372
##  [2,]  2 7.544643
```

```
##  [3,]  3 6.757640
##  [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 5.507210
## [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 8.476700
## [36,] 36 9.604410
## [37,] 37 9.606253
## [38,] 38 5.355979
## [39,] 39 6.917013
## [40,] 40 9.530136
## [41,] 41 9.802424
## [42,] 42 3.891770
## [43,] 43 6.095849
## [44,] 44 9.091670
## [45,] 45 6.737204
## [46,] 46 9.621763
## [47,] 47 9.231489
## [48,] 48 6.404600
## [49,] 49 6.096076
## [50,] 50 8.962319
```

6. Print total number of visits for each `id`.

```
library(plyr)
```

```
##
## Attaching package: 'plyr'
```

```
## The following object is masked from 'package:lubridate':
##
##     here
```

```
totvis <-cbind(unique(xsort[,'id']),count(as.numeric(xsort[,'id']))$freq)
colnames(totvis) <- c('id', 'total visits')
totvis
```

```
##       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
```

7. Print the observations for `id = 15`.

```
xsort[xsort[,'id'] == 15,]
```

```
##      id                  dt      a1c visit
## 11   15 2000-04-30 00:34:50 7.527105     1
## 406  15 2001-01-17 21:11:02 5.898371     2
## 306  15 2001-04-25 06:23:05 8.566593     3
## 152  15 2002-04-25 06:23:05 8.012406     4
## 1522 15 2003-04-25 06:23:05 8.012406     5
## 484  15 2003-06-06 14:06:00 9.133769     6
## 1531 15 2004-06-06 14:06:00 8.012406     7
## 263  15 2004-08-20 17:47:11 8.936190     8
```

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.

```
file3 <- 'https://raw.githubusercontent.com/fonnesbeck/Bios6301/master/datasets/addr.txt'
addr <- readLines(file3)
addr<- gsub('(\\s){2,}', ',', addr)
addr <- gsub('([0-9]{3})(\\s)([A-Z]{1})', '\\1,\\3', addr)
addr <- do.call(rbind, strsplit(addr,','))
colnames(addr) <- c('lastname', 'firstname', 'streetno', 'streetname', 'city', 'state', 'zip')
addr[,7] <- gsub('O', '0', addr[,7]) # Letter 'O' where zeros should be!!!
addr <- gsub('^(\\s)|(\\s)$', '', addr) #remove spaces before and after text
addr
```

```
##       lastname      firstname    streetno streetname
##  [1,] "Bania"       "Thomas M."  "725"    "Commonwealth Ave."
##  [2,] "Barnaby"     "David"      "373"    "W. Geneva St."
##  [3,] "Bausch"      "Judy"       "373"    "W. Geneva St."
##  [4,] "Bolatto"     "Alberto"    "725"    "Commonwealth Ave."
##  [5,] "Carlstrom"   "John"       "933"    "E. 56th St."
##  [6,] "Chamberlin"  "Richard A." "111"    "Nowelo St."
##  [7,] "Chuss"       "Dave"       "2145"   "Sheridan Rd"
##  [8,] "Davis"       "E. J."      "933"    "E. 56th St."
##  [9,] "Depoy"       "Darren"     "174"    "W. 18th Ave."
## [10,] "Griffin"     "Greg"       "5000"   "Forbes Ave."
## [11,] "Halvorsen"   "Nils"       "933"    "E. 56th St."
## [12,] "Harper"      "Al"         "373"    "W. Geneva St."
## [13,] "Huang"       "Maohai"     "725"    "W. Commonwealth Ave."
## [14,] "Ingalls"     "James G."   "725"    "W. Commonwealth Ave."
## [15,] "Jackson"     "James M."   "725"    "W. Commonwealth Ave."
## [16,] "Knudsen"     "Scott"      "373"    "W. Geneva St."
## [17,] "Kovac"       "John"       "5640"   "S. Ellis Ave."
```

```
## [18,] "Landsberg"   "Randy"      "5640"   "S. Ellis Ave."
## [19,] "Lo"          "Kwok-Yung"  "1002"   "W. Green St."
## [20,] "Loewenstein" "Robert F."  "373"    "W. Geneva St."
## [21,] "Lynch"       "John"       "4201"   "Wilson Blvd"
## [22,] "Martini"     "Paul"       "174"    "W. 18th Ave."
## [23,] "Meyer"       "Stephan"    "933"    "E. 56th St."
## [24,] "Mrozek"      "Fred"       "373"    "W. Geneva St."
## [25,] "Newcomb"     "Matt"       "5000"   "Forbes Ave."
## [26,] "Novak"       "Giles"      "2145"   "Sheridan Rd"
## [27,] "Odalen"      "Nancy"      "373"    "W. Geneva St."
## [28,] "Pernic"      "Dave"       "373"    "W. Geneva St."
## [29,] "Pernic"      "Bob"        "373"    "W. Geneva St."
## [30,] "Peterson"    "Jeffrey"    "5000"   "Forbes Ave."
## [31,] "Pryke"       "Clem"       "933"    "E. 56th St."
## [32,] "Rebull"      "Luisa"      "5640"   "S. Ellis Ave."
## [33,] "Renbarger"   "Thomas"     "2145"   "Sheridan Rd"
## [34,] "Rottman"     "Joe"        "8730"   "W. Mountain View Ln"
## [35,] "Schartman"   "Ethan"      "933"    "E. 56th St."
## [36,] "Spotz"       "Bob"        "373"    "W. Geneva St."
## [37,] "Thoma"       "Mark"       "373"    "W. Geneva St."
## [38,] "Walker"      "Chris"      "933"    "N. Cherry St."
## [39,] "Wehrer"      "Cheryl"     "5000"   "Forbes Ave."
## [40,] "Wirth"       "Jesse"      "373"    "W. Geneva St."
## [41,] "Wright"      "Greg"       "791"    "Holmdel-Keyport Rd."
## [42,] "Zingale"     "Michael"    "5640"   "S. Ellis Ave."
##       city         state zip
##  [1,] "Boston"     "MA"  "02215"
##  [2,] "Wms. Bay"   "WI"  "53191"
##  [3,] "Wms. Bay"   "WI"  "53191"
##  [4,] "Boston"     "MA"  "02215"
##  [5,] "Chicago"    "IL"  "60637"
##  [6,] "Hilo"       "HI"  "96720"
##  [7,] "Evanston"   "IL"  "60208-3112"
##  [8,] "Chicago"    "IL"  "60637"
##  [9,] "Columbus"   "OH"  "43210"
## [10,] "Pittsburgh" "PA"  "15213"
## [11,] "Chicago"    "IL"  "60637"
## [12,] "Wms. Bay"   "WI"  "53191"
## [13,] "Boston"     "MA"  "02215"
## [14,] "Boston"     "MA"  "02215"
## [15,] "Boston"     "MA"  "02215"
## [16,] "Wms. Bay"   "WI"  "53191"
## [17,] "Chicago"    "IL"  "60637"
## [18,] "Chicago"    "IL"  "60637"
## [19,] "Urbana"     "IL"  "61801"
## [20,] "Wms. Bay"   "WI"  "53191"
## [21,] "Arlington"  "VA"  "22230"
## [22,] "Columbus"   "OH"  "43210"
## [23,] "Chicago"    "IL"  "60637"
## [24,] "Wms. Bay"   "WI"  "53191"
## [25,] "Pittsburgh" "PA"  "15213"
## [26,] "Evanston"   "IL"  "60208-3112"
## [27,] "Wms. Bay"   "WI"  "53191"
## [28,] "Wms. Bay"   "WI"  "53191"
```

```
## [29,] "Wms. Bay"   "WI"  "53191"
## [30,] "Pittsburgh" "PA"  "15213"
## [31,] "Chicago"    "IL"  "60637"
## [32,] "Chicago"    "IL"  "60637"
## [33,] "Evanston"   "IL"  "60208-3112"
## [34,] "Littleton"  "CO"  "80125"
## [35,] "Chicago"    "IL"  "60637"
## [36,] "Wms. Bay"   "WI"  "53191"
## [37,] "Wms. Bay"   "WI"  "53191"
## [38,] "Tucson"     "AZ"  "85721"
## [39,] "Pittsburgh" "PA"  "15213"
## [40,] "Wms. Bay"   "WI"  "53191"
## [41,] "Holmdel"    "NY"  "07733-1988"
## [42,] "Chicago"    "IL"  "60637"
```

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))))
```

```
##                 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:

```
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.

```
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 first thing that was going wrong was that 'death' was not defined as a global variable. While that did get the function to work, myfun did not return the same output as running the code outside of the function. I determined that the source of the error was not that death was not defined as a global variable, but that the as.formula function wasn't doing exactly what I thought it would do. I found a function to make a variable name a string, and then pasted it to the '~.' before passing it to as.function. This seemed to fix myfun so that I would not need to supply any external information outside of the function to get it to work.

5 bonus points

Create a working function.

```
myfun2 <- function(dat, response) {
  form <- as.formula(paste(substitute(response), '~ .'))
  coef(summary(glm(form , data=dat, family=binomial(logit))))
}
```

```
tryCatch(myfun2(haart_df, death), error = function(e) e)
```

```
##                 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
```