Bios 6301: Assignment 7

Yan Yan

Due Thursday, 08 November, 1:00 PM

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

40 points total.

Submit a single knitr file (named `homework7.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 `homework7.rmd` or include author name may result in 5 points taken off.

Question 1

21 points

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

```r
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: (3 points each)

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

```r
x <- 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.

```r
library(lubridate)
```

```
## Warning: package 'lubridate' was built under R version 3.4.4
```

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

```r
#split x by id
x.list <- split(x,x$id)

#create the new rows and combine them to each list.
x.list.newrow <- lapply(x.list,function(y){
  # diff time between each observations
  diff.y <- difftime(y$dt[2:length(y$dt)], y$dt[1:(length(y$dt)-1)], units='days')
  diff.y.gt1y <- which(diff.y>365)
  if(length(diff.y.gt1y)>0){
    y.newrow <- data.frame(id=y[diff.y.gt1y,]$id,dt=y[diff.y.gt1y,]$dt + years(1),a1c=NA)
    y<-rbind(y,y.newrow)
  }
  y   # return combined or original list
    }
    )

#convert to datafram
x.list.3 <- do.call("rbind",x.list.newrow)

#reorder combined rows
x <- x.list.3[order(x.list.3$id,x.list.3$dt),]
```

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.

```r
# unique id in x

id.x <- table(x$id)
for(i in 1:length(id.x)){
  x[which(x$id == i),'visit'] <- id.x[i]
}
```

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

```r
# calculate all the means for a1c
mean.a1c <- NULL
for(i in 1:length(id.x)){
  mean.a1c[i] <- mean(x[which(x$id == i),]$a1c, na.rm = TRUE )
}


#replace missing values with mean a1c value
for (j in 1:nrow(x)) {
  if(is.na(x$a1c[j])){
    id <- x$id[j]
    x$a1c[j] <- mean.a1c[id]
  }
}
```

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

```r
a1c.sumy <- data.frame(id = as.data.frame(id.x)[,1], a1c = mean.a1c)
a1c.sumy
```

```
##    id      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`.

```r
visit.sum <- as.data.frame(id.x)
colnames(visit.sum)  <- c("id", "visits")
visit.sum
```

```
##    id 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      7
## 16 16      8
## 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`.

```
x[which(x$id == 15),]
```

```
##        id                  dt      a1c visit
## 15.11  15 2000-04-30 00:34:50 7.527105     7
## 15.406 15 2001-01-17 21:11:02 5.898371     7
## 15.306 15 2001-04-25 06:23:05 8.566593     7
## 15.1   15 2002-04-25 06:23:05 8.012406     7
## 15.484 15 2003-06-06 14:06:00 9.133769     7
## 15.2   15 2004-06-06 14:06:00 8.012406     7
## 15.263 15 2004-08-20 17:47:11 8.936190     7
```

Question 2

16 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.

```
# read in the text file and leave the data as strings
addr <- read.delim("addr.txt", header = F, as.is = T)

# strsplit using space >= 2
addr2 <- strsplit(addr[,1],"\\s{2,}")

# test whether I splitted it correctly - each element should have a length of 6
table(unlist(lapply(addr2, length)))
```

```
##
##  6
## 42
```

```
# make it a dataframe
addr3 <- do.call(rbind, addr2)

# remove the space in some zipcodes
addr3[,6] <- gsub("\\s$","",addr3[,6])

# split street no. and street name
str.no <- strsplit(gsub('^([0-9]*) (.*)', '\\1/\\2', addr3[,3]), "/")

# combine as a dataframe
str.info <- do.call(rbind, str.no)

# combine to the other columns
addr4 <- cbind(addr3[,1:2],str.info,addr3[,4:ncol(addr3)])

# names of columns
colnames(addr4) <- c("lastname","firstname","streetno","streetname","city","state","zip")
head(addr4)
```

```
##      lastname     firstname    streetno streetname           city
## [1,] "Bania"      "Thomas M."  "725"    "Commonwealth Ave." "Boston"
## [2,] "Barnaby"    "David"      "373"    "W. Geneva St."     "Wms. Bay"
## [3,] "Bausch"     "Judy"       "373"    "W. Geneva St."     "Wms. Bay"
## [4,] "Bolatto"    "Alberto"    "725"    "Commonwealth Ave." "Boston"
## [5,] "Carlstrom"  "John"       "933"    "E. 56th St."       "Chicago"
## [6,] "Chamberlin" "Richard A." "111"    "Nowelo St."        "Hilo"
##      state zip
## [1,] "MA"  "02215"
## [2,] "WI"  "53191"
## [3,] "WI"  "53191"
## [4,] "MA"  "02215"
## [5,] "IL"  "60637"
## [6,] "HI"  "96720"
```

Question 3

3 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(predvars, data, env): object 'death' not found>
```

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

```
debug(myfun)
```

I added "print(form)" to the line before coef(), and called it again.

```
myfun <- function(dat, response) {
  form <- as.formula(response ~ .)
  print(form)
  coef(summary(glm(form, data=dat, family=binomial(logit))))
}
tryCatch(myfun(haart_df, death), error = function(e) e)
```

```
## response ~ .
## <environment: 0x7fbcac6afe28>

## <simpleError in eval(predvars, data, env): object 'death' not found>
```

It showed that eventhough myfun makes “response ~.” a formula and store it in an object “form”, when printed out, “form” is actually just “response ~ .”, not “death ~ .” as I wanted. So, it failed to pass the true argument to the form.
As traceback records showed, when we call “coef(summary(glm(form, data=haart_df, family=binomial(logit))))”, a serise of functions will be called behind the screen. The error message is created by fucntion 9 to 10 “eval(predvars, data, env)”.

That is, when object “form” was evaluated in glm(), “response” can not be found in the environment, so eval(predvars, data, env) was called to evaluate the object in the parent environment, and got “death”, but “death” is only defined inside the dataframe, so we got the error.
Why “coef(summary(glm(death ~ ., data=haart_df, family=binomial(logit))))” works fine? It’s because “death” can be found in the environment of the datafram where it was defined– haart_df.

5 bonus points

Create a working function.
Based on the above analysis, I first tried to pass the name of the response variable as a string into the argument, and paste it with “~ .” to make a formula.

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

```
##                 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
```

It worked, but you need to quote your response variable.
To omit the “”, I added deparse(substitute(response)) to extract the string, and myfun4 also works, as shown below.

```
myfun4 <- function(dat, response) {
  response <- (deparse(substitute(response)))
  print(response)
  form <- as.formula(paste(response, "~ ."))
  print(form)
  coef(summary(glm(form, data=dat, family=binomial(logit))))
}

myfun4(haart_df, death)
```

```
## [1] "death"
## death ~ .
## <environment: 0x7fbca9fa4268>
```

```
##                 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
```