45

Bios 6301: Assignment 7

Yeji Ko

Due Thursday, 04 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
head(x[order(x$id, x$dt),],10)
```

```
##     id                  dt      a1c
## 32   1 2001-05-08 16:22:52 7.309995
## 268  1 2001-06-17 22:42:23 8.310721
## 201  1 2001-08-17 16:51:46 6.548845
## 285  1 2001-12-14 14:50:29 5.985275
## 194  1 2002-08-19 13:51:47  6.011547
## 304  1 2003-03-22 03:51:36  7.243858
## 493  1 2003-06-27 01:01:34  5.170870
## 464  2 2001-03-05 22:24:43  9.237660
## 286  2 2001-03-16 17:45:49 11.637444
## 133  2 2001-05-02 04:14:56 10.085473
```

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.

```
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':
##
##     between, first, last

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

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

x[, diff := difftime(dt, shift(dt, fill=dt[1L]),
                     units="days"), by=id]

oneyr <- which(x$diff>=365 & x$diff<730)
twoyr <- which(x$diff>=730 & x$diff<1095)

x1 <- x[sort(c(seq_len(nrow(x)), oneyr)), ]
x2 <- x1[sort(c(seq_len(nrow(x1)), twoyr, twoyr)), ]

x2 <- x2 %>% group_by(id) %>% mutate(a1c = replace(a1c, duplicated(diff), NA))

head(x2,10)
```

```
## # A tibble: 10 x 4
## # Groups:   id [2]
##       id dt                    a1c diff
##    <int> <dttm>              <dbl> <drtn>
##  1     1 2001-05-08 16:22:52  7.31   0.00000 days
##  2     1 2001-06-17 22:42:23  8.31  40.26355 days
##  3     1 2001-08-17 16:51:46  6.55  60.75652 days
##  4     1 2001-12-14 14:50:29  5.99 118.95744 days
##  5     1 2002-08-19 13:51:47  6.01 247.91757 days
##  6     1 2003-03-22 03:51:36  7.24 214.62487 days
##  7     1 2003-06-27 01:01:34  5.17  96.84025 days
##  8     2 2001-03-05 22:24:43  9.24   0.00000 days
##  9     2 2001-03-16 17:45:49 11.6   10.80632 days
## 10      2 2001-05-02 04:14:56 10.1   46.39522 days
```

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.

```
x3 <- group_by(x2, id) %>% mutate(visit = row_number())
head(x3,10)
```

```
## # A tibble: 10 x 5
## # Groups:   id [2]
##       id dt                    a1c diff           visit
##    <int> <dttm>              <dbl> <drtn>         <int>
##  1     1 2001-05-08 16:22:52  7.31   0.00000 days     1
##  2     1 2001-06-17 22:42:23  8.31  40.26355 days     2
##  3     1 2001-08-17 16:51:46  6.55  60.75652 days     3
##  4     1 2001-12-14 14:50:29  5.99 118.95744 days     4
##  5     1 2002-08-19 13:51:47  6.01 247.91757 days     5
##  6     1 2003-03-22 03:51:36  7.24 214.62487 days     6
##  7     1 2003-06-27 01:01:34  5.17  96.84025 days     7
##  8     2 2001-03-05 22:24:43  9.24   0.00000 days     1
##  9     2 2001-03-16 17:45:49 11.6   10.80632 days     2
## 10     2 2001-05-02 04:14:56 10.1   46.39522 days     3
```

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

```
x4 <- x3 %>% group_by(id) %>% mutate(a1c= ifelse(is.na(a1c), mean(a1c,na.rm = T),a1c))
head(x4,10)
```

```
## # A tibble: 10 x 5
## # Groups:   id [2]
##       id dt                    a1c diff           visit
##    <int> <dttm>              <dbl> <drtn>         <int>
##  1     1 2001-05-08 16:22:52  7.31   0.00000 days     1
##  2     1 2001-06-17 22:42:23  8.31  40.26355 days     2
##  3     1 2001-08-17 16:51:46  6.55  60.75652 days     3
##  4     1 2001-12-14 14:50:29  5.99 118.95744 days     4
##  5     1 2002-08-19 13:51:47  6.01 247.91757 days     5
##  6     1 2003-03-22 03:51:36  7.24 214.62487 days     6
##  7     1 2003-06-27 01:01:34  5.17  96.84025 days     7
##  8     2 2001-03-05 22:24:43  9.24   0.00000 days     1
##  9     2 2001-03-16 17:45:49 11.6   10.80632 days     2
## 10     2 2001-05-02 04:14:56 10.1   46.39522 days     3
```

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

```
group_by(x4, id) %>% summarise(mean = mean(a1c))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
## # A tibble: 50 x 2
##       id  mean
##    <int> <dbl>
##  1     1  6.65
##  2     2  9.79
##  3     3  6.95
##  4     4  8.19
##  5     5  9.43
## 6      6  7.13
## 7      7  7.88
## 8      8  6.24
## 9      9  4.42
## 10    10  6.03
## # ... with 40 more rows
```

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

```
tally(group_by(x4, id))
```

```
## # A tibble: 50 x 2
##       id     n
##    <int> <int>
##  1     1     7
##  2     2    16
##  3     3    13
##  4     4     9
##  5     5    16
##  6     6    11
##  7     7     5
##  8     8    12
##  9     9    15
## 10    10     8
## # ... with 40 more rows
```

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

```
x4[x4[,'id'] == 15,]
```

```
## # A tibble: 10 x 5
## # Groups:   id [1]
##       id dt                    a1c diff              visit
##    <int> <dttm>              <dbl> <drtn>            <int>
##  1    15 2000-10-21 01:08:17  7.40   0.000000 days     1
##  2    15 2001-08-08 14:23:08  5.90 291.551979 days     2
##  3    15 2001-08-15 07:03:29  7.46   6.694687 days     3
##  4    15 2002-03-15 21:23:10  5.33 212.638669 days     4
##  5    15 2002-04-14 09:08:25  6.48  29.448090 days     5
##  6    15 2002-10-10 18:27:43  8.14 179.388403 days     6
##  7    15 2003-02-19 12:58:53  6.45 131.813310 days     7
##  8    15 2003-03-02 06:58:10  7.43  10.749502 days     8
##  9    15 2003-06-30 07:20:49  7.11 119.974063 days     9
## 10    15 2004-01-22 20:30:42  5.67 206.590197 days    10
```

Question 2

16 points

Install the `lexicon` package. Load the `sw_fry_1000` vector, which contains 1,000 common words.

```
data('sw_fry_1000', package = 'lexicon')
head(sw_fry_1000)
```

```
## [1] "the" "of"  "to"  "and" "a"   "in"
```

1. Remove all non-alphabetical characters and make all characters lowercase. Save the result as `a`.

```r
a <- tolower(gsub("[^[:alpha:] ]",'', sw_fry_1000))
```

Use vector `a` for the following questions. (2 points each)

2. How many words contain the string "ar"?

```r
length(grep('ar',a))
```

```
## [1] 64
```

3. Find a six-letter word that starts with "l" and ends with "r".

```r
a[grep('^l[a-z]{4}r$', a)]
```

```
## [1] "letter"
```

4. Return all words that start with "col" or end with "eck".

```r
a[grep('^col|eck$', a)]
```

```
## [1] "color"   "cold"    "check"   "collect" "colony"  "column"  "neck"
```

5. Find the number of words that contain 4 or more adjacent consonants. Assume "y" is always a consonant.

```r
length(a[grep("[^aeiou]{4,}", a)])
```

```
## [1] 8
```

6. Return all words with a "q" that isn't followed by a "ui".

```r
a[grep("q.[^ui]",a)]
```

```
## [1] "question" "equate"   "square"   "equal"    "quart"    "quotient"
```

7. Find all words that contain a "k" followed by another letter. Run the `table` command on the first character following the first "k" of each word.

```r
a[grep('k[a-z]+', a)]
```

```
##  [1] "like"   "make"   "know"   "take"   "kind"   "keep"   "knew"   "king"
##  [9] "sky"    "kept"   "broke"  "kill"   "lake"   "key"    "skin"   "spoke"
## [17] "skill"  "market"
```

```r
k <- a[grep('k[a-z]+', a)]
table(substr(sub('^[a-z]*(k[a-z]+$)', '\\1', k), 2,2))
```

```
##
##  e  i  n  y
## 10  5  2  1
```

8. Remove all vowels. How many character strings are found exactly once?

```r
sum(nchar(gsub('[aeiou]','', a))==1)
```

```
## [1] 46
```

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.

```r
library(readr)

url <- "https://github.com/couthcommander/Bios6301/raw/master/datasets/haart.csv"
haart <- read_csv("~/Bios6301/datasets/haart.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   male = col_double(),
##   age = col_double(),
##   aids = col_double(),
##   cd4baseline = col_double(),
##   logvl = col_double(),
##   weight = col_double(),
##   hemoglobin = col_double(),
##   init.reg = col_character(),
##   init.date = col_character(),
##   last.visit = col_character(),
##   death = col_double(),
##   date.death = col_character()
## )
```

```r
haart_df <- haart[,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(formula=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(predvars, data, env): object 'death' not found>
```

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

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

```
## debugging in: myfun(haart_df, death)
## debug at <text>#1: {
##     form <- as.formula(response ~ .)
##     coef(summary(glm(formula = form, data = dat, family = binomial(logit))))
## }
## debug at <text>#2: form <- as.formula(response ~ .)
## debug at <text>#3: coef(summary(glm(formula = form, data = dat, family = binomial(logit))))
## <simpleError in eval(predvars, data, env): object 'death' not found>
```

- From debugging, I found out that the data was not called inside the function so the variable 'death' could not be found.

5 bonus points

Create a working function.

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