

Bios 6301: Assignment 7

Lydia Yao

Due Thursday, 04 November, 1:00 PM

$5^{n=\text{day}}$ 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).

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

5. Print mean `a1c` for each `id`.
6. Print total number of visits for each `id`.
7. Print the observations for `id = 15`.

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`.

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

2. How many words contain the string “ar”?
3. Find a six-letter word that starts with “l” and ends with “r”.
4. Return all words that start with “col” or end with “eck”.
5. Find the number of words that contain 4 or more adjacent consonants. Assume “y” is always a consonant.
6. Return all words with a “q” that isn’t followed by a “ui”.
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.
8. Remove all vowels. How many character strings are found exactly once?

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 <- "haart.csv"
haart_df <- read.csv(url)
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.

5 bonus points

Create a working function.