Bios 6301: Assignment 7

Jeongwon Choi

Due Thursday, 03 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.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.4
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.2      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

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`.

```
x <- x[order(x$id, x$dt),]
rownames(x) <- NULL
head(x)
```

	id	dt	a1c
	<int>	<dttm>	<dbl>
1	1	2001-05-08 16:22:52	7.309995
2	1	2001-06-17 22:42:23	8.310721
3	1	2001-08-17 16:51:46	6.548845
4	1	2001-12-14 14:50:29	5.985275

	id	dt	a1c
	<int>	<dttm>	<dbl>
5	1	2002-08-19 13:51:47	6.011547
6	1	2003-03-22 03:51:36	7.243858

6 rows

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.

```
# making a loop that calculates the number of days between the lab test
x$gap <- 0
for (i in 2:nrow(x)) {
  if(x$id[i-1]==x$id[i]){
  x$gap[[i]]=x$dt[i]-x$dt[i-1]
  }
}

# mark refers to the year between the lab tests
x$mark <- x$gap %/% 365
head(x)
```

	id	dt	a1c	gap	mark
	<int>	<dttm>	<dbl>	<dbl>	<dbl>
1	1	2001-05-08 16:22:52	7.309995	0.00000	0
2	1	2001-06-17 22:42:23	8.310721	40.26355	0
3	1	2001-08-17 16:51:46	6.548845	60.75652	0
4	1	2001-12-14 14:50:29	5.985275	118.95744	0
5	1	2002-08-19 13:51:47	6.011547	247.91757	0
6	1	2003-03-22 03:51:36	7.243858	214.62487	0

6 rows

```
#assigning the one year mark to a new row
library(tidyverse)
x$idx <- 1:nrow(x)

head(x)
```

	id	dt	a1c	gap	mark	idx
	<int>	<dttm>	<dbl>	<dbl>	<dbl>	<int>
1	1	2001-05-08 16:22:52	7.309995	0.00000	0	1
2	1	2001-06-17 22:42:23	8.310721	40.26355	0	2
3	1	2001-08-17 16:51:46	6.548845	60.75652	0	3
4	1	2001-12-14 14:50:29	5.985275	118.95744	0	4
5	1	2002-08-19 13:51:47	6.011547	247.91757	0	5
6	1	2003-03-22 03:51:36	7.243858	214.62487	0	6

6 rows

```
nr <-nrow(x)

for (i in 1:nr){
 this_idx <- which(x$idx==i)
 this_mark <- x[this_idx, "mark"]
 if(this_mark==0 ){
   next} else{
for (j in 1:this_mark){

  new.dt <- x[this_idx-1,'dt']+as.difftime(365,units="days")
  x <- x %>% add_row(id=x[this_idx, "id"], dt=new.dt, a1c=NA, .before=this_idx)
  # x[this_idx-1,'gap'] <- x$dt[this_idx-1]-x$dt[this_idx-2]
  x$gap[[this_idx]] =  x$dt[this_idx]-x$dt[this_idx-1]
}
}
}
x
```

id <int>	dt <dttm>	a1c <dbl>	gap <dbl>	mark <dbl>	idx <int>
1	2001-05-08 16:22:52	7.309995	0.000000	0	1
1	2001-06-17 22:42:23	8.310721	40.263553	0	2
1	2001-08-17 16:51:46	6.548845	60.756516	0	3
1	2001-12-14 14:50:29	5.985275	118.957442	0	4
1	2002-08-19 13:51:47	6.011547	247.917569	0	5
1	2003-03-22 03:51:36	7.243858	214.624873	0	6
1	2003-06-27 01:01:34	5.170870	96.840255	0	7
2	2001-03-05 22:24:43	9.237660	0.000000	0	8
2	2001-03-16 17:45:49	11.637444	10.806319	0	9
2	2001-05-02 04:14:56	10.085473	46.395220	0	10

1-10 of 545 rows Previous **1** 2 3 4 5 6 ... 55 Next

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.

```
library(tidyverse)
x$visit <- NULL
uniqueid <- unique(x$id)
for (i in uniqueid) {
  this_id <- which(x$id==i)
  count_this_id <- length(this_id)
  x[this_id,'visit'] <- 1:count_this_id
}
```

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

```
#create a mean a1c value
mean_a1c <- x %>%
group_by(id) %>%
summarize(Mean = mean(a1c, na.rm=TRUE))
mean_a1c
```

id <int>	Mean <dbl>
1	6.654444
2	9.789132
3	6.951820

id <int>	Mean <dbl>
4	8.191985
5	9.429694
6	7.133443
7	7.879138
8	6.244061
9	4.420523
10	6.028370

1-10 of 50 rows Previous **1** 2 3 4 5 Next

```
#replace NA with the mean
uniqueid2 <- unique(x$id)
missingidx <- is.na(x$a1c)

for(i in uniqueid2) {
  this_id2 <- x$id==i
  both <- this_id2 & missingidx
  x[both,'a1c'] <- mean_a1c[i,'Mean']
}
```

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

```
x %>%
group_by(id) %>%
summarize(Mean = mean(a1c))
```

id <int>	Mean <dbl>
1	6.654444
2	9.789132
3	6.951820
4	8.191985
5	9.429694
6	7.133443
7	7.879138
8	6.244061
9	4.420523
10	6.028370

1-10 of 50 rows Previous **1** 2 3 4 5 Next

```
mean_a1c
```

id <int>	Mean <dbl>
1	6.654444
2	9.789132
3	6.951820
4	8.191985
5	9.429694

id <int>	Mean <dbl>
6	7.133443
7	7.879138
8	6.244061
9	4.420523
10	6.028370

1-10 of 50 rows Previous **1** 2 3 4 5 Next

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

```
lastobs <- x %>%
  group_by(id) %>%
  summarise_all(last)

lastobs[,c(1,7)]
```

id <int>	visit <int>
1	7
2	16
3	13
4	9
5	14
6	11
7	7
8	12
9	15
10	8

1-10 of 50 rows Previous **1** 2 3 4 5 Next

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

```
x[x$id == 15,]
```

	id <int>	dt <dttm>	a1c <dbl>	gap <dbl>	mark <dbl>	idx <int>	visit <int>
158	15	2000-10-21 01:08:17	7.401322	0.000000	0	144	1
159	15	2001-08-08 14:23:08	5.896318	291.551979	0	145	2
160	15	2001-08-15 07:03:29	7.457722	6.694687	0	146	3
161	15	2002-03-15 21:23:10	5.330917	212.638669	0	147	4
162	15	2002-04-14 09:08:25	6.484003	29.448090	0	148	5
163	15	2002-10-10 18:27:43	8.139101	179.388403	0	149	6
164	15	2003-02-19 12:58:53	6.446557	131.813310	0	150	7
165	15	2003-03-02 06:58:10	7.432291	10.749502	0	151	8
166	15	2003-06-30 07:20:49	7.113792	119.974063	0	152	9
167	15	2004-01-22 20:30:42	5.668897	206.590197	0	153	10

1-10 of 10 rows

Question 2

16 points

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

```
library(lexicon)
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` .

```
a1 <- tolower(sw_fry_1000)
a <- gsub("[^a-z]", "", a1)
```

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

2. How many words contain the string “ar”?

```
length(grep("ar", a, value=TRUE))
```

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

64 words contain the string “ar”.

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

```
b1 <- grep("^l", a, value=TRUE)
b2 <- grep("r$", b1, value=TRUE)
str_length(b2)
```

```
## [1] 6
```

letter starts with “l” and ends with “r”.

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

```
grep("^col|eck$",a, value=TRUE)
```

```
## [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.

```
length(grep("[^aeiou]{4}", a, value=TRUE))
```

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

8 words contain 4 or more adjacent consonants

6. Return all words with a “q” that isn’t followed by a “ui”.

```
c1 <- grep("[q]", a, value=TRUE)
idx <- grep("[q](!?ui)", c1)
c1[-idx]
```

```
## [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
# I first find words that contain a "k" followed by another letter
d <- grep("[k].+[[:alpha:]]", a, value=TRUE)
# Then split words into a vector of letters
d1 <- strsplit(d, '')
# For vector d1, find where k is located within a word
d2 <- unlist(lapply(d1, function(x){y= x=='k'; return(which(y))}))

# Make a loop with the location of k, return the letter followed by k.
d3 <- c()
for (i in 1:length(d2)){
  d3<- c( d3,d1[[i]][d2[i]+1])
}

# Make a table with d3.
table(d3)
```

```
## d3
## e i n
## 4 5 2
```

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

```r
e1 <- gsub("[aeiou]", '', a)
sum(table(e1)==1)
```

```
## [1] 581
```

581 character strings are found 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. I

```r
# I changed the dataset because error message came out
haart_df <- read.csv('~/downloads/haart.csv')[,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(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.

The 'response' argument in the myfun function is not recognized as an object. So it doesn't work because R cannot recognize the variable such as death or hemoglobin, which makes it impossible to form a new function.

5 bonus points

Create a working function.

```
myfun_new <- function(dat, response) {
  form <- as.formula( paste0(substitute(response), "~.")  )
  coef(summary(glm(form, data=dat, family=binomial(logit))))
}

myfun_new(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
```

So in this case, I included subsitute(response) so that the function could recognize the response variable name as a symbol for formula and use that symbol to create a formula.