Bios 6301: Assignment 7

Yiqing Pan

Due Thursday, 31 October, 1:00 PM

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

40 points total.

Submit a single quarto file (named `homework7.qmd`), 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.qmd` or include author name may result in 5 points taken off.

```
library(dplyr)
```

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

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

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

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)
```

```r
    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, 10)
```

```
##    id                  dt      a1c
## 1  39 2001-11-30 00:36:10 8.758993
## 2  27 2000-07-20 17:05:06 8.233413
## 3  47 2001-02-06 18:40:44 3.950582
## 4  50 2001-04-05 01:55:44 6.707085
## 5  41 2003-08-05 18:30:14 4.047338
## 6  31 2000-12-16 10:38:26 9.357902
## 7  44 2004-01-09 11:17:40 7.967202
## 8  47 2004-10-12 11:27:02 4.559074
## 9  42 2001-10-02 14:04:39 6.932304
## 10 18 2000-08-23 02:44:30 6.122560
```

```r
order_x <- x[order(x$id, x$dt), ]
rownames(order_x) <- 1:nrow(order_x)

head(order_x, 10)
```

```
##    id                  dt       a1c
## 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
## 5   1 2002-08-19 13:51:47  6.011547
## 6   1 2003-03-22 03:51:36  7.243858
## 7   1 2003-06-27 01:01:34  5.170870
## 8   2 2001-03-05 22:24:43  9.237660
## 9   2 2001-03-16 17:45:49 11.637444
## 10  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.

```r
# insert a row after the index indicated
insertRow <- function(data, new_row, index) {
  data_new <- rbind(data[1:index, ],
                    new_row,
```

```r
                    data[- (1:index), ])
  rownames(data_new) <- 1:nrow(data_new)
  return(data_new)
}


current_id = 0
first_test = NA
complete_x <- order_x

for (i in (1: nrow(complete_x))) {
  new_id = complete_x$id[i]
  second_test = complete_x$dt[i]

  if (new_id == current_id) {
    n_gap = as.numeric(second_test - first_test) /365.25

    if (n_gap > 1) {

      for (j in (1:n_gap)) {
        complete_x <- insertRow(complete_x, c(NA, NA, NA), i-1+j-1)
        complete_x$id[i-1+j]=current_id
        complete_x$dt[i-1+j]=first_test+as.difftime(365.25 * j, units = "days")
      }

    }

  }

  current_id = new_id
  first_test = second_test
}

head(complete_x, 50)
```

```
##    id                  dt       a1c
## 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
## 5   1 2002-08-19 13:51:47  6.011547
## 6   1 2003-03-22 03:51:36  7.243858
## 7   1 2003-06-27 01:01:34  5.170870
## 8   2 2001-03-05 22:24:43  9.237660
## 9   2 2001-03-16 17:45:49 11.637444
## 10  2 2001-05-02 04:14:56 10.085473
## 11  2 2001-05-28 12:41:17 11.362266
## 12  2 2001-10-29 11:33:48  8.089224
## 13  2 2001-11-10 11:02:55  9.159491
## 14  2 2002-01-03 05:20:50  7.604405
## 15  2 2002-01-12 04:20:47  8.209176
## 16  2 2003-01-12 10:20:47        NA
## 17  2 2003-06-17 01:43:18  8.743263
```

```
## 18  2 2003-06-26 19:40:59 10.051962
## 19  2 2003-12-05 08:06:49 10.548467
## 20  2 2003-12-28 17:19:13  9.966982
## 21  2 2004-09-19 22:07:42 10.564603
## 22  2 2004-09-20 04:53:12 10.606105
## 23  2 2004-11-27 15:33:28 10.970467
## 24  3 2000-05-01 17:21:57  6.507974
## 25  3 2000-07-04 22:09:43  7.735319
## 26  3 2000-12-24 14:58:33  6.017964
## 27  3 2001-03-29 05:37:39  6.209069
## 28  3 2001-05-26 07:08:17  7.800187
## 29  3 2002-05-26 13:08:17        NA
## 30  3 2002-10-01 08:42:43  6.459650
## 31  3 2003-01-09 11:49:40  8.543998
## 32  3 2004-01-09 17:49:40        NA
## 33  3 2004-01-10 13:37:25 10.047035
## 34  3 2004-03-02 03:03:24  5.551797
## 35  3 2004-06-15 19:14:53  5.541563
## 36  3 2004-07-17 06:47:34  6.055469
## 37  4 2000-05-04 05:40:00  7.892846
## 38  4 2000-06-10 08:40:51  7.871581
## 39  4 2001-03-21 05:55:52  8.264556
## 40  4 2001-08-11 23:41:11  9.045372
## 41  4 2002-02-26 04:44:59  7.255024
## 42  4 2002-09-23 13:23:06  8.667542
## 43  4 2003-09-23 19:23:06        NA
## 44  4 2004-03-12 22:45:37  9.324084
## 45  4 2004-04-24 05:52:05  7.214870
## 46  5 2000-06-03 03:57:21  8.098769
## 47  5 2001-04-07 14:27:32  8.558121
## 48  5 2002-01-13 22:31:45 10.202306
## 49  5 2002-01-15 16:20:01  9.719515
## 50  5 2002-01-21 18:47:11  9.463840
```

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.

```
# id_count <- rep(1, length(unique(complete_x$id)))
#
#  for (i in (1:(nrow(complete_x)-1))) {
#
#    current_id = complete_x$id[i]
#    next_id = complete_x$id[i+1]
#
#    if (current_id == next_id) {
#      id_count[current_id] = id_count[current_id] +1
#    }
#  }
#
# for (i in (1:(nrow(complete_x)))) {
#   current_id = complete_x$id[i]
#   complete_x$visit[i] = id_count[current_id]
# }
```

```r
#
# head(complete_x, 30)


id_count = 1

 for (i in (1:(nrow(complete_x)-1))) {

   current_id = complete_x$id[i]
   next_id = complete_x$id[i+1]

   if (current_id == next_id) {
    complete_x$visit[i] = id_count
    id_count = id_count +1
   } else {
     complete_x$visit[i] = id_count
     id_count = 1
   }
 }

head(complete_x, 30)
```

```
##    id                  dt       a1c visit
## 1   1 2001-05-08 16:22:52  7.309995     1
## 2   1 2001-06-17 22:42:23  8.310721     2
## 3   1 2001-08-17 16:51:46  6.548845     3
## 4   1 2001-12-14 14:50:29  5.985275     4
## 5   1 2002-08-19 13:51:47  6.011547     5
## 6   1 2003-03-22 03:51:36  7.243858     6
## 7   1 2003-06-27 01:01:34  5.170870     7
## 8   2 2001-03-05 22:24:43  9.237660     1
## 9   2 2001-03-16 17:45:49 11.637444     2
## 10  2 2001-05-02 04:14:56 10.085473     3
## 11  2 2001-05-28 12:41:17 11.362266     4
## 12  2 2001-10-29 11:33:48  8.089224     5
## 13  2 2001-11-10 11:02:55  9.159491     6
## 14  2 2002-01-03 05:20:50  7.604405     7
## 15  2 2002-01-12 04:20:47  8.209176     8
## 16  2 2003-01-12 10:20:47        NA     9
## 17  2 2003-06-17 01:43:18  8.743263    10
## 18  2 2003-06-26 19:40:59 10.051962    11
## 19  2 2003-12-05 08:06:49 10.548467    12
## 20  2 2003-12-28 17:19:13  9.966982    13
## 21  2 2004-09-19 22:07:42 10.564603    14
## 22  2 2004-09-20 04:53:12 10.606105    15
## 23  2 2004-11-27 15:33:28 10.970467    16
## 24  3 2000-05-01 17:21:57  6.507974     1
## 25  3 2000-07-04 22:09:43  7.735319     2
## 26  3 2000-12-24 14:58:33  6.017964     3
## 27  3 2001-03-29 05:37:39  6.209069     4
## 28  3 2001-05-26 07:08:17  7.800187     5
## 29  3 2002-05-26 13:08:17        NA     6
## 30  3 2002-10-01 08:42:43  6.459650     7
```

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

```
id_mean <-complete_x %>%
  group_by(id) %>%
  summarize(mean = mean(a1c, na.rm = TRUE)) %>%
  ungroup()

complete_x_woNA <- complete_x


for (i in which(is.na(complete_x_woNA$a1c))) {
  current_id = complete_x_woNA$id[i]

  complete_x_woNA$a1c[i] = id_mean$mean[current_id]

}

head(complete_x_woNA, 50)
```

```
##    id                  dt       a1c visit
## 1   1 2001-05-08 16:22:52  7.309995     1
## 2   1 2001-06-17 22:42:23  8.310721     2
## 3   1 2001-08-17 16:51:46  6.548845     3
## 4   1 2001-12-14 14:50:29  5.985275     4
## 5   1 2002-08-19 13:51:47  6.011547     5
## 6   1 2003-03-22 03:51:36  7.243858     6
## 7   1 2003-06-27 01:01:34  5.170870     7
## 8   2 2001-03-05 22:24:43  9.237660     1
## 9   2 2001-03-16 17:45:49 11.637444     2
## 10  2 2001-05-02 04:14:56 10.085473     3
## 11  2 2001-05-28 12:41:17 11.362266     4
## 12  2 2001-10-29 11:33:48  8.089224     5
## 13  2 2001-11-10 11:02:55  9.159491     6
## 14  2 2002-01-03 05:20:50  7.604405     7
## 15  2 2002-01-12 04:20:47  8.209176     8
## 16  2 2003-01-12 10:20:47  9.789132     9
## 17  2 2003-06-17 01:43:18  8.743263    10
## 18  2 2003-06-26 19:40:59 10.051962    11
## 19  2 2003-12-05 08:06:49 10.548467    12
## 20  2 2003-12-28 17:19:13  9.966982    13
## 21  2 2004-09-19 22:07:42 10.564603    14
## 22  2 2004-09-20 04:53:12 10.606105    15
## 23  2 2004-11-27 15:33:28 10.970467    16
## 24  3 2000-05-01 17:21:57  6.507974     1
## 25  3 2000-07-04 22:09:43  7.735319     2
## 26  3 2000-12-24 14:58:33  6.017964     3
## 27  3 2001-03-29 05:37:39  6.209069     4
## 28  3 2001-05-26 07:08:17  7.800187     5
## 29  3 2002-05-26 13:08:17  6.951820     6
## 30  3 2002-10-01 08:42:43  6.459650     7
## 31  3 2003-01-09 11:49:40  8.543998     8
## 32  3 2004-01-09 17:49:40  6.951820     9
## 33  3 2004-01-10 13:37:25 10.047035    10
## 34  3 2004-03-02 03:03:24  5.551797    11
```

```
## 35  3 2004-06-15 19:14:53  5.541563    12
## 36  3 2004-07-17 06:47:34  6.055469    13
## 37  4 2000-05-04 05:40:00  7.892846     1
## 38  4 2000-06-10 08:40:51  7.871581     2
## 39  4 2001-03-21 05:55:52  8.264556     3
## 40  4 2001-08-11 23:41:11  9.045372     4
## 41  4 2002-02-26 04:44:59  7.255024     5
## 42  4 2002-09-23 13:23:06  8.667542     6
## 43  4 2003-09-23 19:23:06  8.191985     7
## 44  4 2004-03-12 22:45:37  9.324084     8
## 45  4 2004-04-24 05:52:05  7.214870     9
## 46  5 2000-06-03 03:57:21  8.098769     1
## 47  5 2001-04-07 14:27:32  8.558121     2
## 48  5 2002-01-13 22:31:45 10.202306     3
## 49  5 2002-01-15 16:20:01  9.719515     4
## 50  5 2002-01-21 18:47:11  9.463840     5
```

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

```
id_mean$mean
```

```
##  [1]  6.654444  9.789132  6.951820  8.191985  9.429694  7.133443  7.879138
##  [8]  6.244061  4.420523  6.028370  4.838279  6.691181  8.504632  9.122968
## [15]  6.737092  7.420245  6.546329  6.151311  8.628037  8.923518  5.444430
## [22]  5.763931  6.351112  9.377525  5.058097  8.692078  7.371831  4.243469
## [29]  6.345254  4.135795  8.670622  5.130167  6.528153  8.445030  3.832195
## [36]  9.514603  8.612608 10.160773  8.976697  7.583232  3.804325  6.787170
## [43]  5.654235  5.613283  8.876623  7.485824  4.752133  7.415459  5.562809
## [50]  4.970288
```

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

```
id_count <- rep(1, length(unique(complete_x$id)))

 for (i in (1:(nrow(complete_x)-1))) {

   current_id = complete_x$id[i]
   next_id = complete_x$id[i+1]

   if (current_id == next_id) {
     id_count[current_id] = id_count[current_id] +1
   }
 }

id_count
```

```
##  [1]  7 16 13  9 14 11  7 12 15  8 12 12  9 12 10  8 10 14 10 11 13 12 10 12 16
## [26] 11 10 15  3 13 11  9 12 12 11 10  8 14 14 11 14 11  8 12  6 11  9  4 10  7
```

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

```r
complete_x_woNA %>%
  filter(id == 15)
```

```
##    id                  dt      a1c visit
## 1  15 2000-10-21 01:08:17 7.401322     1
## 2  15 2001-08-08 14:23:08 5.896318     2
## 3  15 2001-08-15 07:03:29 7.457722     3
## 4  15 2002-03-15 21:23:10 5.330917     4
## 5  15 2002-04-14 09:08:25 6.484003     5
## 6  15 2002-10-10 18:27:43 8.139101     6
## 7  15 2003-02-19 12:58:53 6.446557     7
## 8  15 2003-03-02 06:58:10 7.432291     8
## 9  15 2003-06-30 07:20:49 7.113792     9
## 10 15 2004-01-22 20:30:42 5.668897    10
```

Question 2

16 points

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

```r
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("[^a-zA-Z]", "", sw_fry_1000))
```

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

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

```r
sum(grepl("ar", a))
```

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

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

```r
a[nchar(a) == 6 & grepl("^l.*r$", a)]
```

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

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

```r
a[grepl("^col", a) | grepl("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
sum(grepl("[bcdfghjklmnpqrstvwxyz]{4}", a, ignore.case = TRUE))
```

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

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

```r
a[grepl("q(?!ui)", a, perl = TRUE)]
```

```
## [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
k_followed <- a[grepl("k[a-zA-Z]", a)]

first_char_after_k <- sapply(k_followed, function(word) {
  # Find the position of k
  pos <- regexpr("k[a-zA-Z]", word)
  if (pos[1] != -1) {
    # Get the character after k
    return(substr(word, pos[1] + 1, pos[1] + 1))
  }
})


table(first_char_after_k)
```

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

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

```r
no_vowels <- gsub("[aeiouAEIOU]", "", a)


unique_count <- table(no_vowels)
exactly_once <- sum(unique_count == 1)
exactly_once
```

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

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
url <- "https://github.com/couthcommander/Bios6301/raw/main/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:

```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 render 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.

In 'as.formula', the function treated response as a string but not a variable that holds the string we want to use. So, when calling myfun(haart_df, death), the response variable name "death" stored in 'response' is not used, the function just use 'response' as the response variable name, which causes error since 'response' is not a variable in the dataset. Also, should use myfun(haart_df, "death") instead of myfun(haart_df, death) as the second argument need to be a string.

5 bonus points

Create a working function.

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

```r
myfun(haart_df, "death") #use string input in the second argument then the function would work
```

```
##                 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
```