+45/40 Awesome!

Bios 6301: Assignment 7

Max Rohde

Great! +21/21

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
# Change name and convert to tibble
df <- tibble(x)

# Order by `id` and `dt`
df %>%
  arrange(id, dt) -> df

df %>%
  head(10) %>%
  kableExtra::kable(booktabs=TRUE)
```

id	dt	a1c
1	2001-05-08 16:22:52	7.309995
1	2001-06-17 22:42:23	8.310721
1	2001-08-17 16:51:46	6.548845
1	2001-12-14 14:50:29	5.985275
1	2002-08-19 13:51:47	6.011547
1	2003-03-22 03:51:36	7.243857
1	2003-06-27 01:01:34	5.170870
2	2001-03-05 22:24:43	9.237660
2	2001-03-16 17:45:49	11.637444
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.

use double pipe

```
# Create a column of difference in years between each timepoint
df %>% %<>%
  group_by(id) %>%
  mutate(difference = c(dt %>% diff() %>% time_length(unit = "year"),0)) %>%
  ungroup() -> df

# Filter only to differences greater than 1 year, and take the floor()
df %>%
  filter(difference >= 1) %>%
  mutate(difference = floor(difference)) -> diffs

# Remove the differences column
df <- df %>% select(-difference)

# Iterate through the differences table
# Add in the missing years
for(i in 1:nrow(diffs)){
  for(j in 1:diffs$difference[i]){
    df <- df %>% add_row(id = diffs$id[i],
                         dt = diffs$dt[i] + years(j),
                         a1c = NA)
  }
}

# Order by `id` and `dt`
df %>%
  arrange(id, dt) -> df

# View first 50 rows
df %>%
  head(50) %>%
  kableExtra::kable(booktabs=TRUE)
```

id	dt	a1c
1	2001-05-08 16:22:52	7.309995
1	2001-06-17 22:42:23	8.310721
1	2001-08-17 16:51:46	6.548845
1	2001-12-14 14:50:29	5.985275
1	2002-08-19 13:51:47	6.011547
1	2003-03-22 03:51:36	7.243857
1	2003-06-27 01:01:34	5.170870
2	2001-03-05 22:24:43	9.237660
2	2001-03-16 17:45:49	11.637444
2	2001-05-02 04:14:56	10.085473
2	2001-05-28 12:41:17	11.362266
2	2001-10-29 11:33:48	8.089225
2	2001-11-10 11:02:55	9.159491
2	2002-01-03 05:20:50	7.604406
2	2002-01-12 04:20:47	8.209176
2	2003-01-12 04:20:47	NA
2	2003-06-17 01:43:18	8.743263
2	2003-06-26 19:40:59	10.051962
2	2003-12-05 08:06:49	10.548466
2	2003-12-28 17:19:13	9.966982
2	2004-09-19 22:07:42	10.564602
2	2004-09-20 04:53:12	10.606105
2	2004-11-27 15:33:28	10.970467
3	2000-05-01 17:21:57	6.507974
3	2000-07-04 22:09:43	7.735319
3	2000-12-24 14:58:33	6.017964
3	2001-03-29 05:37:39	6.209069
3	2001-05-26 07:08:17	7.800187
3	2002-05-26 07:08:17	NA
3	2002-10-01 08:42:43	6.459650
3	2003-01-09 11:49:40	8.543998
3	2004-01-09 11:49:40	NA
3	2004-01-10 13:37:25	10.047035
3	2004-03-02 03:03:24	5.551797
3	2004-06-15 19:14:53	5.541563
3	2004-07-17 06:47:34	6.055469
4	2000-05-04 05:40:00	7.892846
4	2000-06-10 08:40:51	7.871581
4	2001-03-21 05:55:52	8.264556
4	2001-08-11 23:41:11	9.045372
4	2002-02-26 04:44:59	7.255024
4	2002-09-23 13:23:06	8.667542
4	2003-09-23 13:23:06	NA
4	2004-03-12 22:45:37	9.324084
4	2004-04-24 05:52:05	7.214870
5	2000-06-03 03:57:21	8.098769
5	2001-04-07 14:27:32	8.558121
5	2002-01-13 22:31:45	10.202306
5	2002-01-15 16:20:01	9.719515
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.

```
df %>%
  group_by(id) %>%   n()
  mutate(visit = 1:length(id)) -> df

df %>%
  head(20) %>%
  kableExtra::kable(booktabs=TRUE)
```

id	dt	a1c	visit
1	2001-05-08 16:22:52	7.309995	1
1	2001-06-17 22:42:23	8.310721	2
1	2001-08-17 16:51:46	6.548845	3
1	2001-12-14 14:50:29	5.985275	4
1	2002-08-19 13:51:47	6.011547	5
1	2003-03-22 03:51:36	7.243857	6
1	2003-06-27 01:01:34	5.170870	7
2	2001-03-05 22:24:43	9.237660	1
2	2001-03-16 17:45:49	11.637444	2
2	2001-05-02 04:14:56	10.085473	3
2	2001-05-28 12:41:17	11.362266	4
2	2001-10-29 11:33:48	8.089225	5
2	2001-11-10 11:02:55	9.159491	6
2	2002-01-03 05:20:50	7.604406	7
2	2002-01-12 04:20:47	8.209176	8
2	2003-01-12 04:20:47	NA	9
2	2003-06-17 01:43:18	8.743263	10
2	2003-06-26 19:40:59	10.051962	11
2	2003-12-05 08:06:49	10.548466	12
2	2003-12-28 17:19:13	9.966982	13

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

```
df %>%
  group_by(id) %>%
  mutate(a1c = ifelse(is.na(a1c), mean(a1c,na.rm=TRUE),a1c)) -> df
```

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

```
df %>%
  group_by(id) %>%
  summarise(mean_a1c = mean(a1c, na.rm = TRUE)) %>%
  kableExtra::kable(booktabs=TRUE)
```

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

id	mean_a1c
1	6.654444
2	9.789132
3	6.951821
4	8.191985
5	9.429694
6	7.133444
7	7.879138
8	6.244061
9	4.420523
10	6.028370
11	4.838279
12	6.691181
13	8.504632
14	9.122968
15	6.737092
16	7.420245
17	6.546329
18	6.151311
19	8.628037
20	8.923518
21	5.444430
22	5.763931
23	6.351112
24	9.377525
25	5.058097
26	8.692078
27	7.371831
28	4.243468
29	6.345254
30	4.135795
31	8.670622
32	5.130167
33	6.528153
34	8.445030
35	3.832195
36	9.514603
37	8.612608
38	10.160773
39	8.976697
40	7.583232
41	3.804325
42	6.787170
43	5.654235
44	5.613283
45	8.876624
46	7.485824
47	4.752133
48	7.415459
49	5.562809
50	4.970288

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

```
df %>%
  group_by(id) %>%
  summarise(num_visit = max(visit)) %>%
  kableExtra::kable(booktabs=TRUE)
```

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

id	num_visit
1	7
2	16
3	13
4	9
5	14
6	11
7	7
8	12
9	15
10	8
11	12
12	12
13	9
14	12
15	10
16	8
17	10
18	14
19	10
20	11
21	13
22	12
23	10
24	12
25	16
26	11
27	10
28	15
29	3
30	13
31	11
32	9
33	12
34	12
35	11
36	10
37	8
38	14
39	14
40	11
41	14
42	11
43	8
44	12
45	6
46	12
47	10
48	5
49	11
50	9

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

```
df %>%
  filter(id == 15) %>%
  kableExtra::kable(booktabs=TRUE)
```

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

Question 2

Great! +16/16

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`.

```
sw_fry_1000 %>%
  str_replace_all("\\W", "") %>%
  str_to_lower() -> a

# Examine the differences
sw_fry_1000[sw_fry_1000 != a]
```

```
## [1] "I"     "don't" "won't"
```

```
a[sw_fry_1000 != a]
```

```
## [1] "i"    "dont" "wont"
```

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

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

```
str_detect(a, "ar") %>% sum()
```

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

```
# We see that 64 words contain the string "ar"
```

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

```
a[str_detect(a, "^l.{4}r$")]
```

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

```
# The only match is "letter"
```

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

```
a[str_detect(a, "(^col|eck$)")]
```

```
## [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.

```
str_detect(a, "[^aeiou]{4,}") %>% sum()
```

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

```
# 8 words fit this description
```

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

```
a[str_detect(a, "q(?!ui)")]
```

```
## [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.

```
a[str_detect(a, "k.")]
```

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

```
str_match(a, "k(.)")[,2] %>% table()
```

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

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

```
no_vowel <- str_replace_all(a, "[aeiou]","")

(table(no_vowel) == 1) %>% sum()
```

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

```
# There are 581 strings 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 <- "https://github.com/couthcommander/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.

Answer The issue is that when `death` is passed into the function, R tries to interpret it as an object coming from the global namespace, where it doesn't exist, since it only exists in relation to the data frame.

5 bonus points

Create a working function.

```
# Here is a working version using reformulate()
myfun <- function(dat, response) {
  form <- reformulate(".", deparse(substitute(response)))
  coef(summary(glm(form, data=dat, family=binomial(logit))))
}

myfun(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
```