Problem Set #03 Solutions Statistics 506

Problem Set #03

Problem 1 Solutions - Vision

The complete Do-file can be found here.

a.

```
. import sasxport5 "https://wwwn.cdc.gov/Nchs/Nhanes/2005-2006/VIX_D.XPT", clear
. rename _all, lower
  (all newnames==oldnames)
. quietly compress
. save ~/Desktop/vision, replace
file ~/Desktop/vision.dta saved
. import sasxport5 "https://wwwn.cdc.gov/Nchs/Nhanes/2005-2006/DEMO_D.XPT", clear
. rename _all, lower
  (all newnames==oldnames)
. quietly compress
. merge 1:1 seqn using ~/Desktop/vision
    Result                      Number of obs
    -----------------------------------------
    Not matched                         3,368
        from master                     3,368  (_merge==1)
        from using                          0  (_merge==2)

    Matched                             6,980  (_merge==3)
    -----------------------------------------
. keep if _merge == 3
(3,368 observations deleted)
. count
  6,980
```

b.

Clean up and rename variables.

```
. rename viq220 glasses
. replace glasses = . if glasses == 9
(2 real changes made, 2 to missing)
. replace glasses = glasses - 1
(6,545 real changes made)
. rename ridageyr age
```

Generate a variable capturing the age brackets.

```
. capture drop agecat
. generate agecat = floor(age/10)
. replace agecat = . if missing(age)
(0 real changes made)
```

I'll demonstrate a number of different approaches:

1

Using `mean`

Since the average of a binary variable is it's proportion, I can use `mean` to generate the proportions, then just need to clean up the matrix it produces.

```
. capture matrix drop prop
. quietly mean glasses, over(agecat)
. matrix prop = e(b)'
. matrix prop = prop*100
. matrix colnames prop = "Proportion"
. matrix rownames prop = "10-19" "20-29" "30-39" "40-49" "50-59" ///
>                        "60-69" "70-79" "80-89"
. matrix list prop, format(%3.1f)
prop[8,1]
       Proportion
10-19        67.9
20-29        67.3
30-39        64.1
40-49        63.0
50-59        45.0
```

```
60-69        37.8
70-79        33.1
80-89        33.1
```

2

Using `tabulate` and `mata` to manually calculate

`tabulate` generates a table with the count of 0's and 1's in glasses. I can use `mata` to quickly calculate proportions, then again clean up the matrix.

```
. capture matrix drop prop
. tabulate agecat glasses, matcell(x)
           |    Glasses/contact
           |    lenses worn for
           |       distance
    agecat |         0          1 |     Total
-----------+----------------------+----------
         1 |       670      1,418 |     2,088
         2 |       306        631 |       937
         3 |       269        481 |       750
         4 |       286        487 |       773
         5 |       335        274 |       609
         6 |       392        238 |       630
         7 |       299        148 |       447
         8 |       208        103 |       311
-----------+----------------------+----------
     Total |     2,765      3,780 |     6,545
. mata:
------------------------------------------------- mata (type end to exit) ----------
: x = st_matrix("x")
: x = x[., 2]:/(x[., 1] + x[., 2])
: st_matrix("x", x)
: end
------------------------------------------------------------------------------------
. matrix prop = x*100
. matrix colnames prop = "Proportion"
. matrix rownames prop = "10-19" "20-29" "30-39" "40-49" "50-59" ///
>                        "60-69" "70-79" "80-89"
. matrix list prop, format(%3.1f)
prop[8,1]
```

```
       Proportion
10-19        67.9
20-29        67.3
30-39        64.1
40-49        63.0
50-59        45.0
60-69        37.8
70-79        33.1
80-89        33.1
```

3

Using `table`

`table` is designed for this sort of operation, though it's harder if not impossible to format things as nicely.

```
. table (agecat) (), stat(mean glasses) nototal nformat("%9.3f" mean)
----------------
        |   Mean
--------+-------
agecat  |
  10-19 |  0.679
  20-29 |  0.673
  30-39 |  0.641
  40-49 |  0.630
  50-59 |  0.450
  60-69 |  0.378
  70-79 |  0.331
  80-89 |  0.331
----------------
```

4

Using a loop to fill a matrix

This is the most R-like approach; generating an empty matrix and filling it up as we loop through the categories. Finally, we again clean up the matrix prior to displaying.

```
. capture matrix drop prop
. matrix define prop = (0\0\0\0\0\0\0\0)
. foreach n of numlist 1/8 {
  2.         local lowerage = `n'*10
  3.         local upperage = (`n'+1)*10
  4.         quietly mean glasses if age >= `lowerage' & age < `upperage'
  5.         matrix prop[`n', 1] = e(b)
  6. }
. matrix prop = prop*100
. matrix colnames prop = "Proportion"
. matrix rownames prop = "10-19" "20-29" "30-39" "40-49" "50-59" ///
>                        "60-69" "70-79" "80-89"
. matrix list prop, format(%3.1f)
prop[8,1]
       Proportion
10-19        67.9
20-29        67.3
30-39        64.1
40-49        63.0
50-59        45.0
60-69        37.8
70-79        33.1
80-89        33.1
```

5

Calculate in-place in the dataset

We can use `collapse` to replace our dataset with the proportions we want, do some minor cleaning, then just print the data.

```
. preserve
. collapse (mean) glasses, by(agecat)
. rename agecat Age
. rename glasses Percent
. replace Percent = Percent*100
(8 real changes made)
. format Percent %9.1f
. list, sep(0)

     +-----------------+
```

```
     |   Age   Percent |
     |-----------------|
  1. | 10-19      67.9 |
  2. | 20-29      67.3 |
  3. | 30-39      64.1 |
  4. | 40-49      63.0 |
  5. | 50-59      45.0 |
  6. | 60-69      37.8 |
  7. | 70-79      33.1 |
  8. | 80-89      33.1 |
     +-----------------+
. restore
```

c.

Clean up and rename variables.

```
. rename riagendr gender
.
. capture drop female
. generate female = gender == 2
. replace female = . if missing(female)
(0 real changes made)
.
. rename ridreth1 race
. label define race 1 "Mexican American" ///
>                   2 "Other Hispanic" ///
>                   3 "Non-Hispanic White" ///
>                   4 "Non-Hispanic Black" ///
>                   5 "Multi-racial"
. label values race race
.
. rename indfmpir pir
```

Run the models.

```
. quietly logit glasses c.age
. estimate store m1
. quietly logit glasses c.age i.race i.female
. estimate store m2
. quietly logit glasses c.age i.race i.female c.pir
. estimate store m3
```

Produce the table.

```
. estimates table m1 m2 m3, stats(N r2_p aic) eform


-------------------------------------------------
    Variable |     m1          m2          m3
-------------+-----------------------------------
         age |  .97562897    .97767872     .978056
             |
        race |
Other His..  |               .8552837    .89045517
Non-Hispa..  |              .51225603    .60560404
Non-Hispa..  |               .7696096    .81270706
Multi-rac~l  |              .52152821      .587002
             |
      female |
           1  |              .60526462    .59674151
             |
         pir |                           .89261693
       _cons |  3.5288428    6.2755778    7.5094302
-------------+-----------------------------------
           N |       6545         6545         6247
        r2_p |  .04973123    .07195445    .07339952
         aic |  8475.8866    8287.7609    7909.8082
-------------------------------------------------
```

d.

```
. estimates restore m3
(results m3 are active now)
. logit, or

Logistic regression                                     Number of obs =  6,247
                                                        LR chi2(7)    = 625.30
                                                        Prob > chi2   = 0.0000
Log likelihood = -3946.9041                             Pseudo R2     = 0.0734

------------------------------------------------------------------------------
     glasses | Odds ratio   Std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
         age |    .978056   .0012665   -17.14   0.000     .9755769    .9805414
```

```
                 |
            race |
  Other Hispanic |   .8904552   .1498325    -0.69   0.490     .6403015    1.238339
Non-Hispanic W.. |    .605604   .0455103    -6.67   0.000     .5226635    .7017062
Non-Hispanic B.. |   .8127071   .0643806    -2.62   0.009     .6958313    .9492139
    Multi-racial |    .587002   .0822693    -3.80   0.000     .4460076    .7725683
                 |
        1.female |   .5967415    .032406    -9.51   0.000     .5364902    .6637595
             pir |   .8926169   .0158059    -6.42   0.000     .8621694    .9241397
           _cons |    7.50943   .6592368    22.97   0.000     6.322398    8.919328
------------------------------------------------------------------------------
Note: _cons estimates baseline odds.
```

The estimated odds ratio for females is ~.60 and statistically significant, providing evidence that the odds of females wearing glasses/contacts for distance vision is statistically significantly lower than the odds for males.

```
. margins female

Predictive margins                                      Number of obs = 6,247
Model VCE: OIM

Expression: Pr(glasses), predict()

------------------------------------------------------------------------------
             |            Delta-method
             |     Margin   std. err.      z    P>|z|     [95% conf. interval]
-------------+----------------------------------------------------------------
      female |
          0  |   .6334955   .0083353    76.00   0.000     .6171587    .6498324
          1  |   .5190438   .0084379    61.51   0.000     .5025058    .5355819
------------------------------------------------------------------------------
. margins female, pwcompare(pv)

Pairwise comparisons of predictive margins              Number of obs = 6,247
Model VCE: OIM

Expression: Pr(glasses), predict()

------------------------------------------------------
             |            Delta-method    Unadjusted
             |   Contrast   std. err.      z    P>|z|
```

```
-------------+----------------------------------------
      female |
     1 vs 0  |  -.1144517   .0118695    -9.64   0.000
-------------------------------------------------------
```

We also see evidence that females have a statistically significantly lower probability of wearing glasses/contact lenses for distance vision than males.

Problem 2 Solutions - Salika

```
library(DBI)
sakila <- dbConnect(RSQLite::SQLite(), "data/sakila_master.db")
```

a.

```
dbGetQuery(sakila, "
SELECT l.name, count(l.name) AS count
  FROM film AS f
  LEFT JOIN language AS l on f.language_id = l.language_id
 GROUP BY l.name
 ORDER by -count
")
```

```
     name count
1 English  1000
```

Trick question: They're all in English.

b.

R approach:

```
fc <- dbGetQuery(sakila, "SELECT * FROM film_category")
cat <- dbGetQuery(sakila, "SELECT * FROM category")
catcount <- table(fc$category_id)
maxcat <- which.max(catcount)
c(cat$name[cat$category_id == maxcat], catcount[maxcat])
```

```
              15
"Sports"     "74"
```

With a single query:

```
dbGetQuery(sakila, "
SELECT c.name, count(c.category_id) AS count
  FROM category as c
 RIGHT JOIN film_category AS fc ON fc.category_id = c.category_id
 GROUP BY c.category_id
 ORDER by -count
 LIMIT 1
")
```

```
    name count
1 Sports    74
```

c.

First, to use R, let's extract all relevant tables into data.frames.

```
customer <- dbGetQuery(sakila, "SELECT * FROM customer")
address  <- dbGetQuery(sakila, "SELECT * FROM address")
city     <- dbGetQuery(sakila, "SELECT * FROM city")
country  <- dbGetQuery(sakila, "SELECT * FROM country")
```

Next, we can use merges to mimic what SQL is actually doing. (This produces a warning, but on variables we don't care about)

```
merged1 <- merge(customer, address, by = "address_id",
                 all.x = TRUE, all.y = FALSE)
merged2 <- merge(merged1, city, by = "city_id",
                 all.x = TRUE, all.y = FALSE)
merged3 <- merge(merged2, country, by = "country_id",
                 all.x = TRUE, all.y = FALSE)
```

```
Warning in merge.data.frame(merged2, country, by = "country_id", all.x = TRUE,
: column names 'last_update.x', 'last_update.y' are duplicated in the result
```

```
t <- table(merged3$country)
t[t == 9]
```

```
United Kingdom
             9
```

Here's a more R-style approach:

```
cities <- address$city_id[match(customer$address_id, address$address_id)]
countries <- city$country_id[match(cities, city$city_id)]
tcountries <- table(country$country[match(countries, country$country_id)])
tcountries[tcountries == 9]
```

```
United Kingdom
             9
```

Finally, the query:

```
dbGetQuery(sakila, "
SELECT co.country, count(co.country) AS count
  FROM country AS co
 RIGHT JOIN
    (SELECT country_id
       FROM city AS ci
      RIGHT JOIN
         (SELECT city_id
            FROM customer AS c
            LEFT JOIN address AS a ON c.address_id = a.address_id
         ) AS ca ON ca.city_id = ci.city_id
    ) AS ccc ON ccc.country_id = co.country_id
 GROUP BY co.country
HAVING count == 9")
```

```
         country count
1 United Kingdom     9
```

Problem 3

```
dat <- read.csv("data/us-500.csv")
```

a.

```
length(dat$email[grepl("net$", dat$email)])/nrow(dat)
```

```
[1] 0.14
```

b.

Checking the username portion first - extract the usernames, then detect anything non-alphanumeric

```
emails <- strsplit(dat$email, "@")
usernames <- sapply(emails, "[[", 1)
username_non_alphanumeric <- grepl("[^a-zA-Z0-9]", usernames)
```

Repeat for domains, stripping off the TLD first.

```
domains <- sapply(emails, "[[", 2)
domains <- gsub("\\.[a-z]{3}", "", domains)
domain_non_alphanumeric <- grepl("[^a-zA-Z0-9]", domains)
```

Finally, we can get the proportion.

```
mean(username_non_alphanumeric | domain_non_alphanumeric)
```

```
[1] 0.506
```

c.

First, make sure that all the phone numbers are the same number of digits so we don't have preceeding 1's or anything like that

```
table(sapply(dat$phone1, nchar))
```

```
 12
500
```

Looks good, so we can assume the first three characters of every number is the area code.

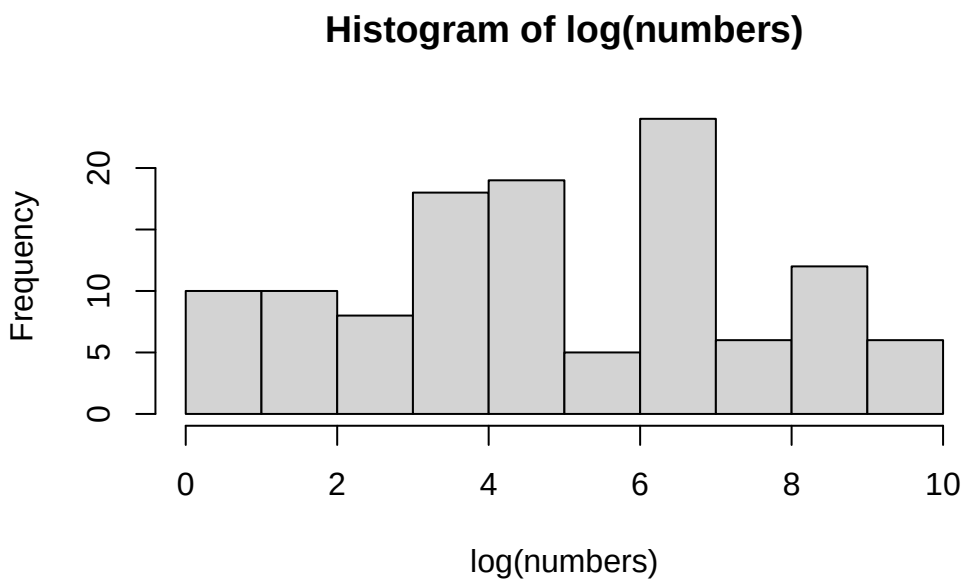
```
phone1area <- substr(dat$phone1, 1, 3)
phone2area <- substr(dat$phone2, 1, 3)
sort(table(c(phone1area, phone2area)), decreasing = TRUE)[1]
```

```
973
 36
```

d.

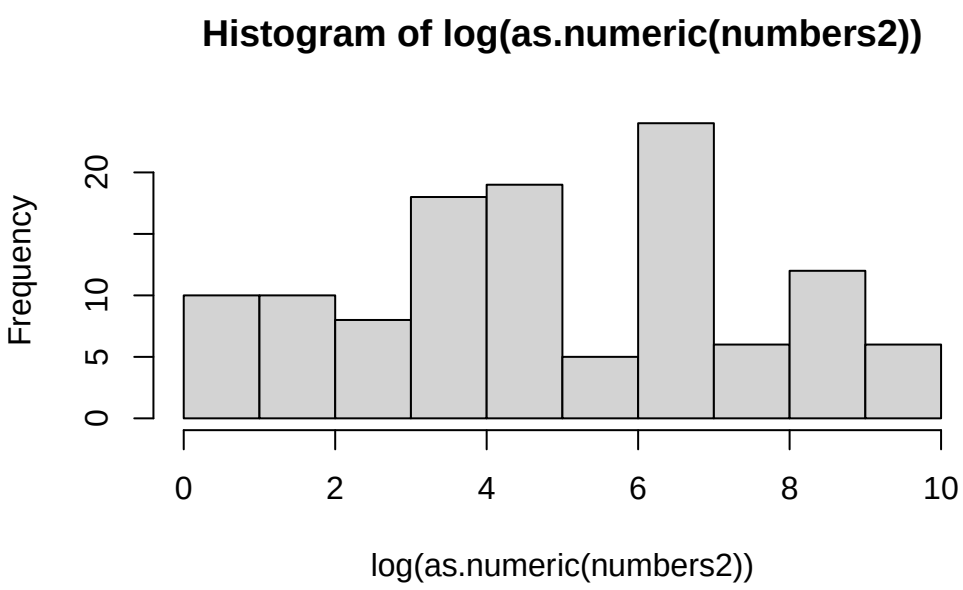
In this first approach, we identify any address that ends in a number, then split the string on spaces and store the last entry.

```
apartments <- dat$address[grepl("[0-9]+$", dat$address)]
numbers <- sapply(strsplit(apartments, " "), function(x) x[length(x)])
numbers <- as.numeric(gsub("#", "", numbers))
hist(log(numbers))
```



Another approach; here we just extract out the numbers that are at the end of the address.

```
numbers2 <- regmatches(apartments, regexpr("[0-9]+$", apartments))
hist(log(as.numeric(numbers2)))
```



Note that for this problem, there were edge cases that would require human intervention to uncover. For example,

```
apartments[108]
```

```
[1] "51120 State Route 18"
```

Here the "18" is the highway number, not an apartment number. Identifying all cases where this occurs would require full knowledge of all such possible roads - e.g. "State Route", "Highway", "Rt", "Route", etc. In addition, this assumes no user-error on input. Generally these would require a human review of the input or a far more advanced solution. This is true of most human-readble data, and especially true of user-input data. Handling this is not required for this problem.

e.

```
table(substr(numbers, 1, 1))
```

```
 1  2  3  4  5  6  7  8  9
15 13 12 12 15 11 12 11 17
```

This is a uniform distribution, rather than the decreasing distribution as expected by Benford's law. This data obviously does not appear real.

f.

```
housenumbers <- sapply(strsplit(dat$address, " "), function(x) x[1])
lastnum <- sapply(housenumbers, function(x) {
  substr(x, length(x), length(x))
})
table(lastnum)
```

```
lastnum
 1  2  3  4  5  6  7  8  9
52 63 67 58 43 55 60 48 54
```

We see another uniform distribuion. It gets a bit tricky to apply Benford's law. On non-leading digits, Benford's law predicts a uniform distribution as the position of the digit increases. However, this is even trickier in this case as the last digit does not hold a fixed position:

```
table(sapply(housenumbers, nchar))
```

```
  1   2   3   4   5
106 115  82 109  88
```

We can look at each length separately.

```
lens <- sapply(housenumbers, nchar)
for (l in names(table(lens))) {
  print(table(lastnum[lens == l]))
}
```

```
 1  2  3  4  5  6  7  8  9
12 13  9 16 13 10 11  8 14

 1  2  3  4  5  6  7  8  9
10 10 17 12  8 15 18 13 12

 1  2  3  4  5  6  7  8  9
```

```
10  9 12  9 10  8 12  5  7

 1  2  3  4  5  6  7  8  9
11 23 20 12  3  9  8 13 10

 1  2  3  4  5  6  7  8  9
 9  8  9  9  9 13 11  9 11
```

We seem, if anything, an increase in the frequency of the larger digits with low lengths. But overall, all look like noisy uniforms distributions. So on the one hand, with large counts this supports Benford's Law, more realisitically, the first position results (either from part e. or the first table when length = 1) being uniform contradicts Benford's Law for this data, as expected by the artificial nature of this data.