HW2

AnyaConti28661255

February 13, 2017

EXPLAIN 3

1.a. Loading the data into a dataframe called ca_pa

```
ca_pa <- read.csv("/Users/Anya/Documents/JuniorYear/Spring/Stat 597A/calif_penn_2011.csv")
```

1.b. Obtained the dimensions for the dataframe with the command dim(). The data set has 11275 rows and 34 columns.

```
dim(ca_pa)
```

```
## [1] 11275    34
```

1.c. This command sums up the number of blank cells in each column.

```
colSums(apply(ca_pa,c(1,2),is.na))
```

```
##                             X                        GEO.id2
##                             0                              0
##                       STATEFP                       COUNTYFP
##                             0                              0
##                       TRACTCE                     POPULATION
##                             0                              0
##                      LATITUDE                      LONGITUDE
##                             0                              0
##             GEO.display.label            Median_house_value
##                             0                            599
##                   Total_units                  Vacant_units
##                             0                              0
##                  Median_rooms    Mean_household_size_owners
##                           157                            215
## Mean_household_size_renters           Built_2005_or_later
##                           152                             98
##            Built_2000_to_2004                   Built_1990s
##                            98                             98
##                   Built_1980s                   Built_1970s
##                            98                             98
##                   Built_1960s                   Built_1950s
##                            98                             98
##                   Built_1940s         Built_1939_or_earlier
##                            98                             98
##                    Bedrooms_0                    Bedrooms_1
##                            98                             98
##                    Bedrooms_2                    Bedrooms_3
##                            98                             98
##                    Bedrooms_4            Bedrooms_5_or_more
##                            98                             98
##                        Owners                        Renters
##                           100                            100
##       Median_household_income         Mean_household_income
```

```
##                 115                 126
```

1.d. Used na.omit() to eliminate any rows in the data set with blank data, and assigned this to a new data set called newca_pa. Note: the pre-written code on the homework following this question is changed to refer to the new data set.

```
newca_pa <- na.omit(ca_pa)
```

1.e. There were 670 rows eliminated. This was figured out by subtracted the row dimension of the new data set newca_pa from the row dimension of the original data set ca_pa.

```
dim(ca_pa)[1] - dim(newca_pa)[1]
```

```
## [1] 670
```

1.f. The answers to c and e are indeed compatible. Though summing the total number of blank cells per column, and thus the number of blank cells overall comes to 3034 (see first line of code below) which is well over the number of rows removed (670), this makes sense because there are probably some rows with multiple blank cells. In addition, the highest number of blank cells in a column was 599 (second row of code below), and we know at least all of those are from separate rows, and so the fact that the number of rows removed is higher than that also makes sense makes sense.

```
sum(colSums(apply(ca_pa,c(1,2),is.na)))
```

```
## [1] 3034
```

```
max(colSums(apply(ca_pa,c(1,2),is.na)))
```

```
## [1] 599
```

2.a. (Note: Copied code but changed ca_pa to newca_pa.) First a blank vector acca (named for Alameda County, CA) is created. Then a for-loop is created where the variable "tract"" holds the place of the row number. If that particular row of the state column has the value 6 and the row of the county column has the value 1, then that row number is added to acca. By the end of the for loop, acca will be a complete list of all the row numbers that have 6 for state, and 1 for county. Then a blank vector accamhv (mhv stands for median housing value) is created. Then another for-loop where the variable "tract"" holds the place, this time taking on the values of acca one at a time, which still represent particular rows in the data frame newca_pa. The median housing value of that particular row is then added to accamhv (10th column, row represented by tract). After this loop is over, the median of all those values is then taken.

```
acca <- c()
for (tract in 1:nrow(newca_pa)) {
  if (newca_pa$STATEFP[tract] == 6) {
    if (newca_pa$COUNTYFP[tract] == 1) {
      acca <- c(acca, tract)
    }
  }
}
accamhv <- c()
for (tract in acca) {
  accamhv <- c(accamhv, newca_pa[tract,10])
}
median(accamhv)
```

```
## [1] 474050
```

2.b. Takes the median of the Median House value column, but only for the rows where the state code is 6 (CA) and the county code is 1 (Alameda County).

```
median(newca_pa$Median_house_value[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==1])
```

```
## [1] 474050
```

2.c. Here are the average percentages of housing buit in 2005 or later in each of the 3 counties specified. **2.c.i.** Takes the mean of percent of houses built 2005 or later column for the rows where the state code is 6 (CA) and the county code is 1 (Alameda County)

```
mean(newca_pa$Built_2005_or_later[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==1])
```

```
## [1] 2.820468
```

2.c.ii. Takes the mean of percent of houses built 2005 or later column for the rows where the state code is 6 (CA) and the county code is 85 (Santa Clara County)

```
mean(newca_pa$Built_2005_or_later[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==85])
```

```
## [1] 3.200319
```

2.c.iii. Takes the mean of percent of houses built 2005 or later column for the rows where the state code is 42 (PA) and the county code is 3 (Allegheny County)

```
mean(newca_pa$Built_2005_or_later[newca_pa$STATEFP==42 & newca_pa$COUNTYFP==3])
```

```
## [1] 1.474219
```

2.d. Finding correlations between median housing value and percent of houses built in 2005 or later in various categories: **2.d.i.** Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later, for all data

```
cor(x = newca_pa$Median_house_value, y = newca_pa$Built_2005_or_later )
```

```
## [1] -0.01893186
```

2.d.ii. Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later, only when the state code in each is 6 (CA)

```
cor(x = newca_pa$Median_house_value[newca_pa$STATEFP==6], y = newca_pa$Built_2005_or_later[newca_pa$STA
```

```
## [1] -0.1153604
```

2.d.iii. Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later only when the state code in each is 42 (PA)

```
cor(x = newca_pa$Median_house_value[newca_pa$STATEFP==42], y = newca_pa$Built_2005_or_later[newca_pa$ST
```

```
## [1] 0.2681654
```

2.d.iv. Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later only when the state code in each is 6 (CA) and the county code in each is 1 (Alameda County)

```
cor(x = newca_pa$Median_house_value[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==1], y = newca_pa$Built_200
```

```
## [1] 0.01303543
```

2.d.v. Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later only when the state code in each is 6 (CA) and the county code in each is 85 (Santa Clara County)

```
cor(x = newca_pa$Median_house_value[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==85], y = newca_pa$Built_20
```

```
## [1] -0.1726203
```

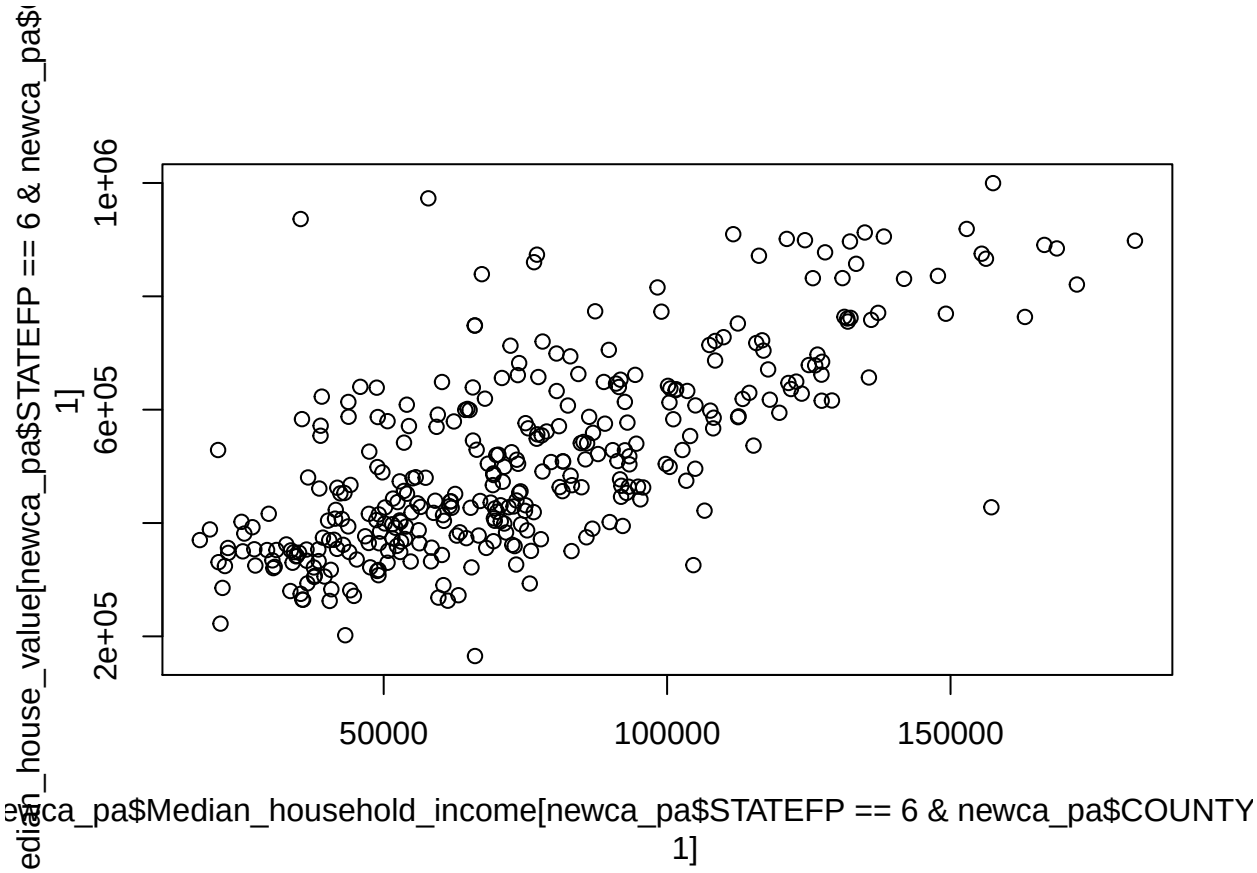
2.d.vi. Uses the cor() function to find the correlation coefficient between the median housing value and the percent of houses built in 2005 or later only when the state code in each is 3 (PA) and the county code in each is 3 (Allegheny County)

```
cor(x = newca_pa$Median_house_value[newca_pa$STATEFP==42 & newca_pa$COUNTYFP==3], y = newca_pa$Built_20
```

```
## [1] 0.1939652
```

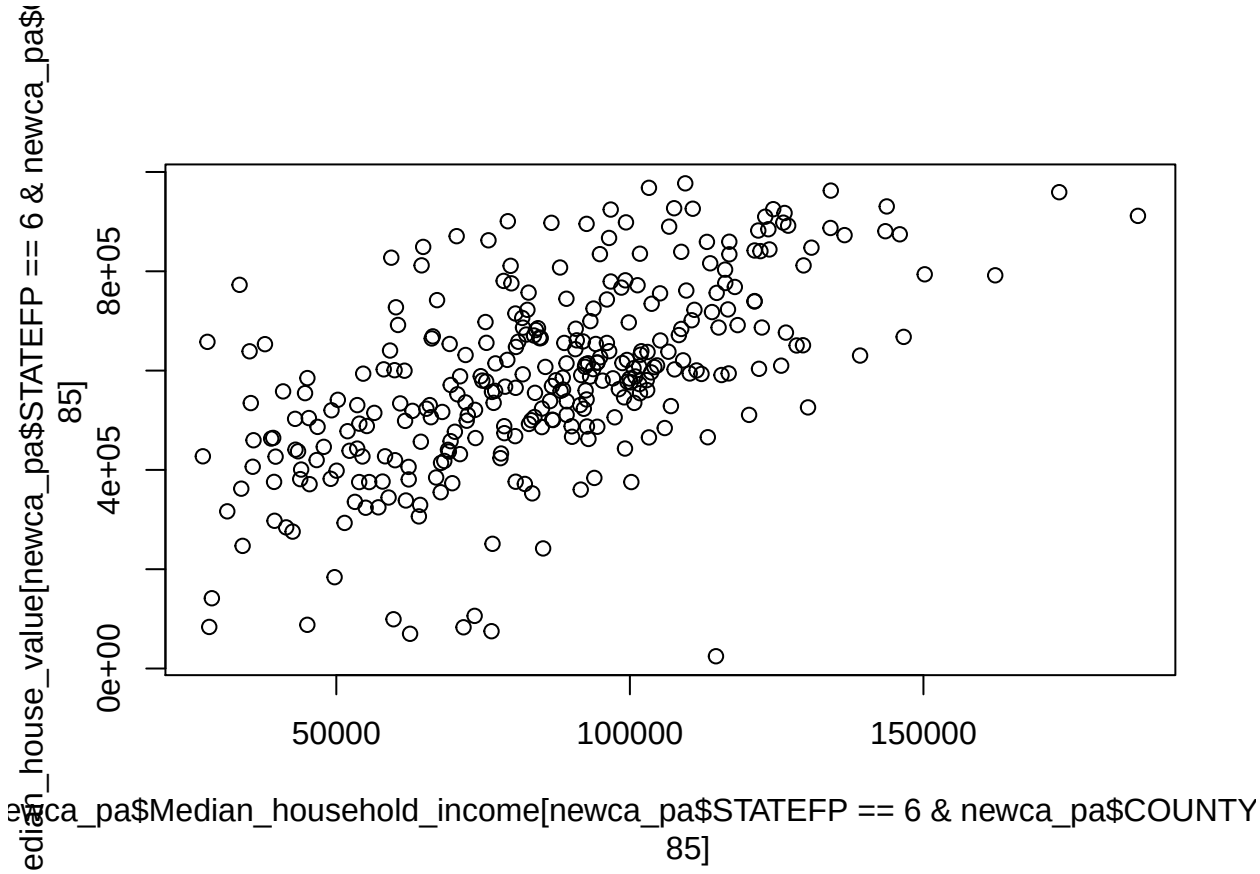
2.e. For each of the counties given, plots the median house value against median income **2.e.i.** Uses the plot() function to plot the median house value against median income for Alameda county (by only taking rows where the state code is 6 and the county code is 1)

```
plot(y = newca_pa$Median_house_value[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==1], x = newca_pa$Median_h
```



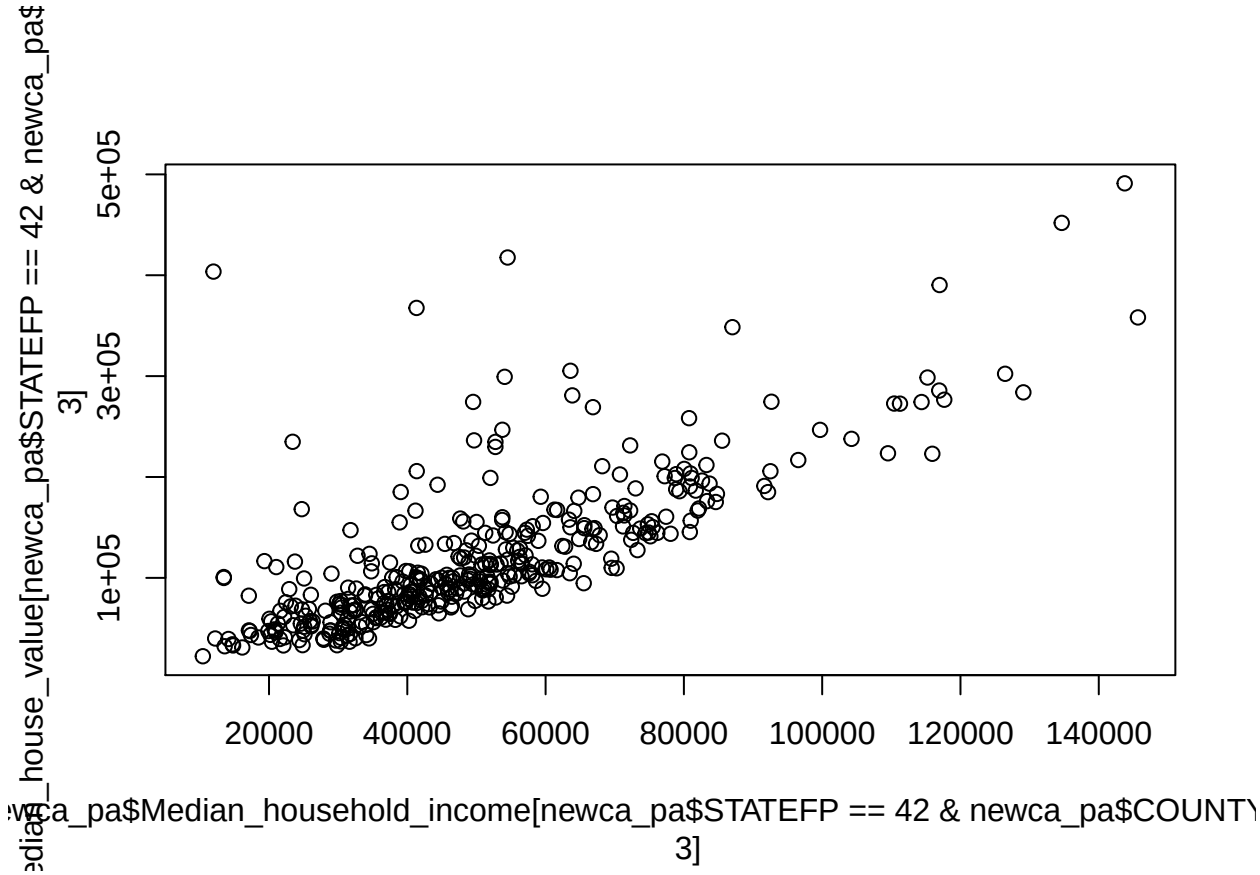
2.e.ii. Uses the plot() function to plot the median house value against median income for Santa Clara county (by only taking rows where the state code is 6 and the county code is 85)

```
plot(y = newca_pa$Median_house_value[newca_pa$STATEFP==6 & newca_pa$COUNTYFP==85], x = newca_pa$Median_
```



2.e.iii. Uses the plot() function to plot the median house value against median income for Allegheny county (by only taking rows where the state code is 42 and the county code is 3)

```
plot(y = newca_pa$Median_house_value[newca_pa$STATEFP==42 & newca_pa$COUNTYFP==3], x = newca_pa$Median_
```



3.a. First the data is read into a new data frame called gmpdata using the command read.table(). Then in the next line, a new column is created for the population, which is gotten by dividing the gmp by pcgmp (pcgmp is per capita gmp, or gmp/population). Then a function called mse is created with 3 arguments: the first is called y0a (assumed to be a vector with 2 values), the second is called N and is set to have a default value of the pop column of gmpdata, and the third is called Y and is set to have a default value of the pcgmp column of gmpdata. (Note: mse can still be run with different N and Y data by specifying inputs such as mle(c(y0,a),new N, new Y) but if they are left blank, then the ones specified are the default values such as in mle(c(y0,a)). Then the mean squared error is calculated based on the equation provided. Y is represented by the pcgmp data, N is represented by the pop data, y0 is represented by the first value of y0a, a is represented by the second value of y0a, and n is represented by the length of the data set (either N or Y, but in this case N). Then the function is run, putting in a vector for y0a with two values of 6611 and 0.15 to test it. This is then repeated with the values 5000 and 0.10. The output matches what should happen.

```
gmpdata <- read.table("http://people.math.umass.edu/~wei/Teaching/STAT597_Spring17/gmp.dat")

gmpdata$pop <- gmpdata$gmp / gmpdata$pcgmp

mse <- function(y0a, N = gmpdata$pop, Y = gmpdata$pcgmp){
  (1/length(N))*sum((Y-y0a[1] * (N^y0a[2]))^2)
}

mse(c(6611,0.15))
```

```
## [1] 207057513
```

```
mse(c(5000,0.10))
```

```
## [1] 298459915
```

3.b. The nlm() function is run below, eithe the mse as its first argument, and the vector of starting values (y0a) as it's second argument. This is run with 3 different starting pairs in the vector y0a, the first and second number of each pair representing y0 and a respectively. Minimum represents the estimated minimum value of the function, and estimate represents the estimated values of intputs (y0 and a) at which that minimum is obtained.

```
nlm(mse,c(6611,0.15))
```

```
## $minimum
## [1] 61857061
##
## $estimate
## [1] 6610.9999997    0.1263182
##
## $gradient
## [1]   51.76342 -210.18945
##
## $code
## [1] 2
##
## $iterations
## [1] 7
```

```
nlm(mse,c(5000,0.10))
```

```
## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value

## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value

## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value

## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value

## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value

## Warning in nlm(mse, c(5000, 0.1)): NA/Inf replaced by maximum positive
## value
```

```
## $minimum
## [1] 62521485
##
## $estimate
## [1] 5000.0000008    0.1475913
##
## $gradient
## [1] -1028.22561    11.38855
##
## $code
## [1] 2
##
## $iterations
```

```
## [1] 5
```

```
nlm(mse,c(7000,0.20))
```

```
## $minimum
## [1] 1168662933
##
## $estimate
## [1] 6999.9983 -153.6302
##
## $gradient
## [1] 0 0
##
## $code
## [1] 1
##
## $iterations
## [1] 1
```

3.c. A function called plm is created, with 3 different arguments, the same ones as used in the mse function from 3.a. The first is called y0a (assumed to be a vector with 2 values), the second is called N, and the third is called Y. The default value for N is set the same as mse: the population column of gmpdata, but this can be overridden by putting a different input vector in while running the function. The default value of Y is set the same as mse as well: the per capita gmp of gmpdata, but this can be overridden by putting a different input vector in while running the function. Thus the function runs nlm() on the function mse, using inputs for mse created as a vector made of the first input of y0a (representing y0), and the second imput of y0a (representing a) put together, as well as an N Vector, and a Y vector. The results of nlm() are then saved to the variable minEst. Then a list called output is created consisting of the final guess for y0 (stored as the first entry in the estimate section of minEst which is the nlm results), the final guess for a (stored as the second entry in the estimate section of minEst which is the nlm results), and the final value of the MSE function (stored as the first entry of the minimum section of minEst which is the nlm results). plm() is then run with an input of y0=6611 and a=0.15, and then again with y0=5000 and a=0.10. For each, the parameter estimates for y0 were very close to the original starting values. For the first however, the parameter estimate for a went down to around 0.126 while for the second it went up to 1.476. They differ because this only provides an estimate, and finds it based on the starting values. The one with the lower mean square error is the first one.

```
plm <- function(y0a, N=gmpdata$pop, Y=gmpdata$pcgmp){
  minEst <- nlm(mse,c(y0a[1],y0a[2]), N, Y)
  output <- c("Final Guess for y0" = minEst$estimate[1], "Final Guess for a" = minEst$estimate[2] ,"Min
  return(output)
}

plm(c(6611,0.15))
```

```
## Final Guess for y0  Final Guess for a  Min Mean Sq Error
##       6.611000e+03       1.263182e-01       6.185706e+07
```

```
plm(c(5000,0.10))
```

```
## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
```

```
## positive value

## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
## positive value

## Warning in nlm(mse, c(y0a[1], y0a[2]), N, Y): NA/Inf replaced by maximum
## positive value

## Final Guess for y0  Final Guess for a  Min Mean Sq Error
##       5.000000e+03       1.475913e-01       6.252148e+07
```