PPAS Challenge Answers

Contents

Intro

Here we present potential answers to the challenge questions provided. There is almost always more than one way to perform even simple tasks in R, so consider these merely suggested answers. Recall that we used the USArrests and state.x77 datasets from R's "datasets" package. This package should already be loaded into your console when you open R Studio. The questions can be found in the accompanying Word document.

Notes:

- *Not all output is shown because some of it is big and cumbersome. We encourage you to run these lines of code on your own machine.*
- *We use the function kable() from the "knitr" package to clean up the output of tables, but this is more important for displaying output in a document like this. The kable() function is hardly necessary for displaying tables in your own console.*

Load packages

Load necessary packages.

```
# install.packages("datasets"); library(datasets) just in case you don't have it!
library(dplyr)
library(car)
library(knitr)
```

Questions

Data summary (Question 1)

Q1: Take a look at R's documentation of these datasets to familiarize yourself with them. Look at data summaries and histograms to get a sense for the distribution of values. Are both datasets of the class "data.frame"? You'll probably want to make sure they both are.

A1: Often when working with large datasets, this step can catch obvious outliers and data errors. Additionally, you will surely find it useful to know more about the distributions of your relevant variables.

```
?USArrests
?state.x77
```

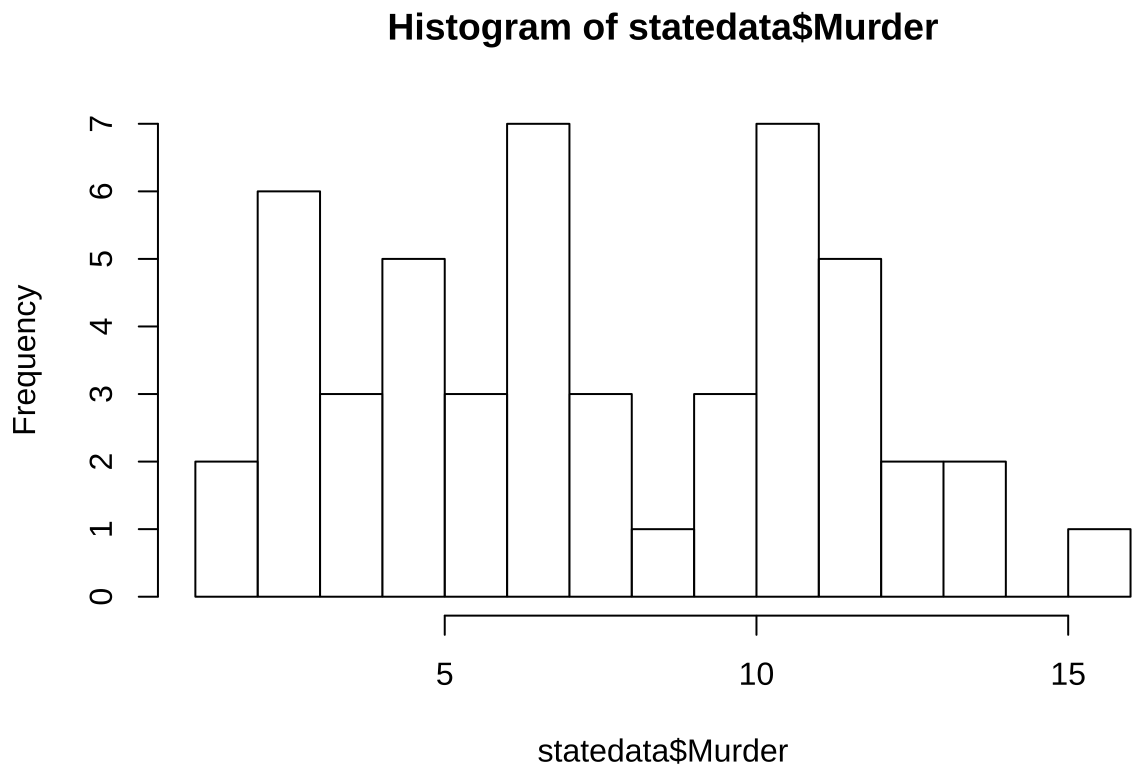
```r
summary(USArrests)
```

```
     Murder          Assault         UrbanPop          Rape      
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00  
```

```r
summary(state.x77)
```

```
   Population        Income       Illiteracy       Life Exp    
 Min.   :  365   Min.   :3098   Min.   :0.500   Min.   :67.96  
 1st Qu.: 1080   1st Qu.:3993   1st Qu.:0.625   1st Qu.:70.12  
 Median : 2838   Median :4519   Median :0.950   Median :70.67  
 Mean   : 4246   Mean   :4436   Mean   :1.170   Mean   :70.88  
 3rd Qu.: 4968   3rd Qu.:4814   3rd Qu.:1.575   3rd Qu.:71.89  
 Max.   :21198   Max.   :6315   Max.   :2.800   Max.   :73.60  
     Murder          HS Grad          Frost             Area       
 Min.   : 1.400   Min.   :37.80   Min.   :  0.00   Min.   :  1049  
 1st Qu.: 4.350   1st Qu.:48.05   1st Qu.: 66.25   1st Qu.: 36985  
 Median : 6.850   Median :53.25   Median :114.50   Median : 54277  
 Mean   : 7.378   Mean   :53.11   Mean   :104.46   Mean   : 70736  
 3rd Qu.:10.675   3rd Qu.:59.15   3rd Qu.:139.75   3rd Qu.: 81163  
 Max.   :15.100   Max.   :67.30   Max.   :188.00   Max.   :566432  
```

```r
statedata <- as.data.frame(state.x77)
hist(statedata$Murder,
     breaks = 10)
```



And a ggplot solution:

```
library(ggplot2)
ggplot(statedata,
       mapping = aes(x = Murder)) +
  geom_histogram(binwidth = 1)
```

Data prep (Questions 2 - 3)

Q2: Join the information from the two data frames together into a single data frame, matching by state.

A2: Here we use more than one method to join data, left_join and cbind.

```
arrestdata <- USArrests %>%
  mutate(State = rownames(USArrests))
statedata <- statedata %>%
  mutate(State = rownames(statedata))
joindata.1 <- statedata %>%
  left_join(arrestdata, by = c("State" = "State"))
joindata.2 <- cbind(state.x77, USArrests)
```

Q3: You'll want to make sure that the names of your columns make sense, and that no two columns have the same name.

A3: In joindata.2, note that there are two "Murder" columns with the same name. This could be a problem if we were to use that version of the joined data from the cbind function, so we'll proceed with the first dataset, joindata.1. We clarify below which murder rates are which by manipulating the data frame's names. We also show a new input to the left_join function that adds suffixes to column names in the intersection of both data frames.

```
names(joindata.1)[names(joindata.1) %in% c("Murder.x", "Murder.y")] <- c("Murder1976", "Murder1973")
joindata.1 <- statedata %>%
  left_join(arrestdata, by = c("State" = "State"),
            suffix = c("1976", "1973"))
```

Clean up the workspace, one file at a time.

```
rm(arrestdata, joindata.2, statedata)
gc()
```

This is a quick way to remove all but a few objects. The gc() function helps to actually clear out the objects and the RAM they are using.

```
rm(list = ls()[!(ls() %in% c("joindata.1"))])
gc()
```

Analysis (Questions 4 - 7)

Q4: Create a correlation matrix of all the numeric columns. Later in modeling, it will be important to know which variables are correlated to each other.

A4: You might be tempted to make a correlation matrix using a double loop. Please resist. Note that you must remove any categorical variables from the data frame before using the cor() function.

```
cor(joindata.1 %>%
      select(-State))
```

Q5: Create a pivot table that splits observations into five groups of ten, ordered by 1973 murder rates, and then calculates average 1976 murder rates within each group.

A5: This just one instance where dplyr comes in very handy. Pivot tables are easy! Note that you could have created a group column for 1973 murder rates in a separate mutate step beforehand, but here I did it all in one step using ntile. ntile() allows you to bucket continuous variables into equally sized groups, sorted in numerical order.

```
kable(joindata.1 %>%
  group_by(MRate1973.ntile = ntile(Murder1973, 5)) %>%
  summarize(MRate1973 = mean(Murder1973),
            MRate1976 = mean(Murder1976)), digits = 2)
```

MRate1973.ntile	MRate1973	MRate1976
1	2.38	3.34
2	4.77	4.24
3	7.21	6.82
4	10.28	10.21
5	14.30	12.28

Q5 (bonus): If you're feeling especially bold, pick three new variables from the dataset, split each one into two groups by ordered values (ntile), and then calculate average 1976 murders rates in each of the eight group combinations.

A5 (bonus): To pivot over more variables, it's as simple as inputting more into the group_by() function. Note that it is also possble, and probably more common even, to group by existing categorical variables without needing to bucket them.

```
kable(joindata.1 %>%
  group_by(Illiteracy.ntile = ntile(Illiteracy, 2),
           UrbanPop.ntile = ntile(UrbanPop, 2),
           Mrate1973.ntile = ntile(Murder1973, 2)) %>%
  summarize(MRate1976 = mean(Murder1976),
            Obs = n()),
  digits = 2)
```

Illiteracy.ntile	UrbanPop.ntile	Mrate1973.ntile	MRate1976	Obs
1	1	1	3.92	13
1	2	1	4.44	5
1	2	2	9.27	7
2	1	1	6.70	1
2	1	2	11.81	11
2	2	1	4.58	6
2	2	2	9.54	7

Not all groupings are guaranteed to have the same sample size, and in fact, one potential grouping had no observations. Nowhere was there an above average murder rate with below average illiteracy and below average urban population (the 1,1,2 combo). Note that we used the n() function to count observations in each grouping.

Q6: Create a linear regression model to predict murder rates in 1976 using information from previous years. Feel free to use any predictor variables that make sense, but be sure to include murder and assault rates in 1973 in order to answer later parts of this question.

Q6a: Notice that two of the column names from the original state.x77 dataset have spaces. This creates problems in fitting a linear model if you want to use those variables. Change those variable names so that they don't have spaces.

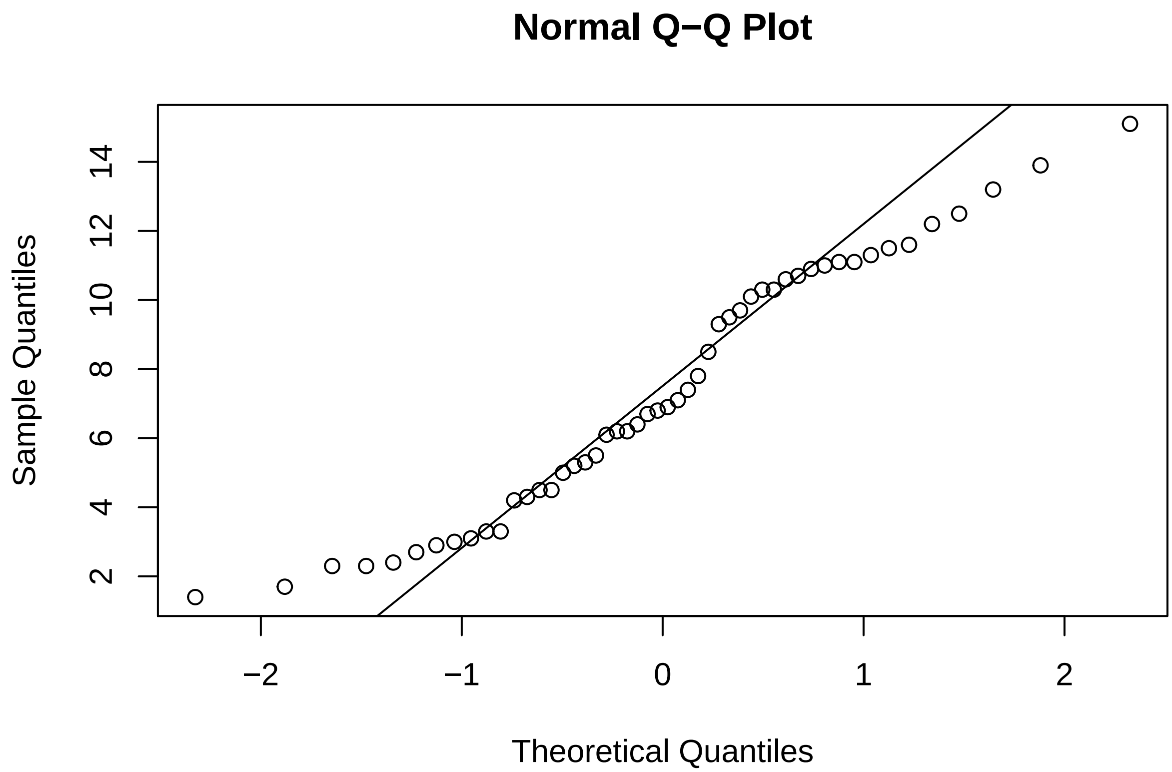
A6a: "Life Exp" and "HS Grad" were column names that came from a matrix where that type of chicanery is allowed. We'll rename those now, and note that you can always index things numerically if it's easier, as we've done below.

```
names(joindata.1)[c(4, 6)] <- c("LifeExp", "HSGrad")
```

Q6b: Check the normality of the response variable using a quantile-quantile plot, and/or find a statistical hypothesis test for normality.

A6b: Below we check the normality of the 1976 murder rates by state, showing both a q-q plot and the results of the Shipiro-Wilk test for normality.

```
qqnorm(joindata.1$Murder1976)
qqline(joindata.1$Murder1976)
```




```
shapiro.test(joindata.1$Murder1976)
```

```

	Shapiro-Wilk normality test

data:  joindata.1$Murder1976
W = 0.95347, p-value = 0.04745
```

The response data appear to have light tails based on the quantile-quantile plot, and the Shapiro-Wilk test rejects the normality assumption. So we will proceed with caution as we model. Our p-values have the potential to be misleading.

Q6c: Fit a model to predict murder rates by state in 1976, using at least 1973's assault and murder rates, and then anything else you think might be predictive. Look at the model summary.

A6c: We fit a linear model to predict 1976 murder rates. Note that the I() function allows you to mutate new variables within the modeling step. Here I have derived each state's population density from the Population and Area variables and included it as a predictor in the model.

```
model.1 <- lm(Murder1976 ~ Murder1973 + Assault +
                UrbanPop + I(Population/Area) +
                Illiteracy + Income + HSGrad,
              data = joindata.1)
summary(model.1)
```

```

Call:
lm(formula = Murder1976 ~ Murder1973 + Assault + UrbanPop + I(Population/Area) +
    Illiteracy + Income + HSGrad, data = joindata.1)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-2.4912 -0.8106 -0.2244  0.8384  3.1844 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)         0.3180039  2.3043478   0.138   0.8909    
Murder1973          0.6825086  0.1039134   6.568 6.05e-08 ***
Assault             0.0007798  0.0042603   0.183   0.8556    
UrbanPop            0.0070789  0.0198220   0.357   0.7228    
I(Population/Area) -2.6356262  1.3168199  -2.002   0.0518 .  
Illiteracy          0.7699857  0.5127795   1.502   0.1407    
Income              0.0003847  0.0004802   0.801   0.4275    
HSGrad             -0.0200828  0.0485994  -0.413   0.6815    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.319 on 42 degrees of freedom
Multiple R-squared:  0.8906,    Adjusted R-squared:  0.8724 
F-statistic: 48.86 on 7 and 42 DF,  p-value: < 2.2e-16
```

Past murder rates seem to predict future murder rates well. But it seems weird at first that, despite a 0.74 correlation coefficient between Assault and Murder1976, past assault rates are not a statistically signficant predictor of future murder rates in this model. What we haven't taken into account is the high linear correlation coefficient between the 1973 assault and murder rates predictor variables. The coefficients in linear models are more interpretable when the variables have low correlations. In this case, the 1973 murder rates variable is the stronger predictor, and it has stolen any thunder that past assault rates may have had.

Q7: Note the high correlation between assault rates in 1973 and murder rates in 1973 in your correlation matrix from earlier. One of those variables is likely to be statistically insignificant in your linear model.

Q7a: Think about what is happening here, and what we can do to clarify effects in a linear model. Implement your idea as part of your best model.

A7a: We know that when predictor variables are highly correlated, interpretation of the coefficients and their significance can become difficult. Let's try a little trick, and then we'll also remove some of the least significant variables in the model (based solely on p-values, for now).

```r
cor(joindata.1$Murder1973, joindata.1$Assault)
```

```
[1] 0.8018733
```

```r
cor(joindata.1$Murder1973, joindata.1$Assault/joindata.1$Murder1973)
```

```
[1] -0.5301196
```

```r
model.2 <- lm(Murder1976 ~ Murder1973 + I(Assault/Murder1973) +
                I(Population/Area) + Illiteracy,
              data = joindata.1)
kable(summary(model.2)$coef, digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.5543	0.8458	1.8377	0.0727
Murder1973	0.7133	0.0659	10.8206	0.0000
I(Assault/Murder1973)	-0.0073	0.0221	-0.3288	0.7438
I(Population/Area)	-1.8466	0.8539	-2.1625	0.0359

	Estimate	Std. Error	t value	Pr(>|t|)
Illiteracy	0.6190	0.4283	1.4452	0.1553

By using the ratio of assault rates to murder rates, we are able to reduce the absolute correlation between the two model variables from 0.80 to 0.53, while retaining some unique, potentially predictive information in the assault rates. As it turns out, the assault rate variable is still statistically insignificant as you can see above. But in general, using ratios and differences between variables can help to extract information from more predictors without clogging the model with variables that are too linearly correlated. A more relevant example might be predicting variable annuity lapse rates from knowledge of account value (AV) and benefit base (BB). These are typically very correlated; however, we can use AV to represent policy size and BB/AV to represent in-the-moneyness in a linear model and avoid multicollinearity.

Q7b: Arrive at a best model, and check the residual plots for any funny business.

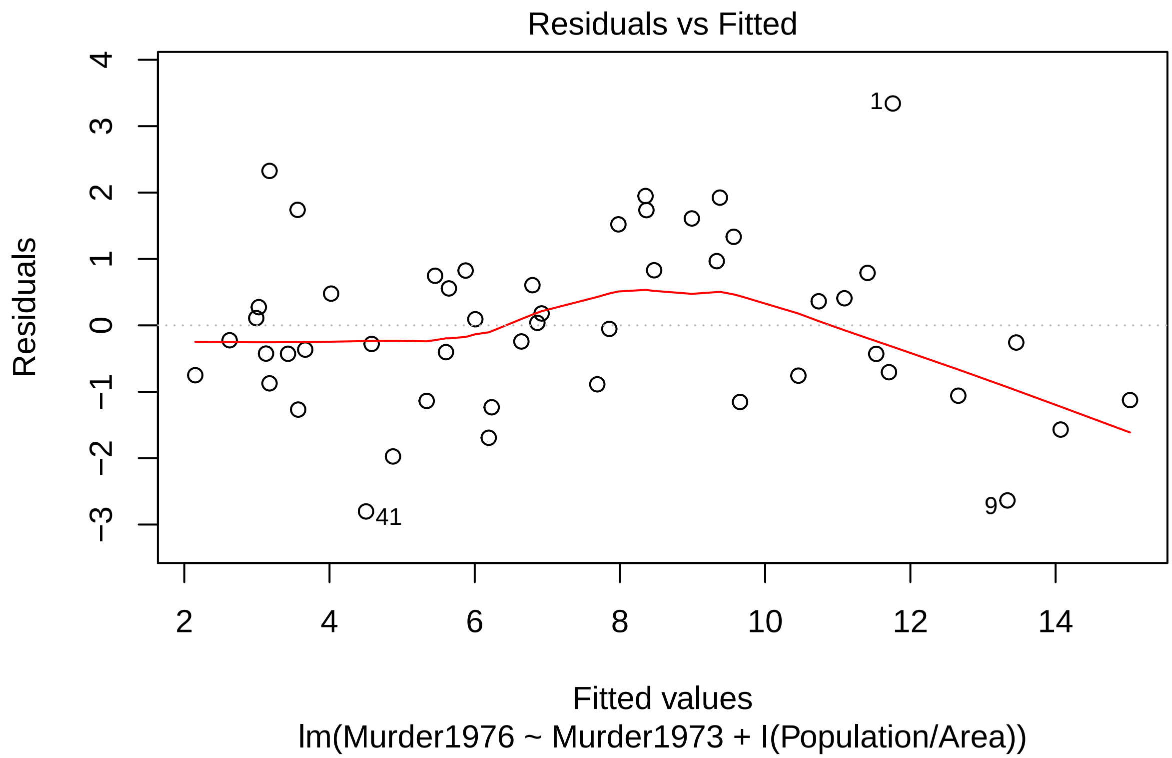
A7b: Here's my final model, after removing variables/coefficients with high p-values.

```
model.3 <- lm(Murder1976 ~ Murder1973 + I(Population/Area), data = joindata.1)
kable(summary(model.3)$coef, digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.5404	0.4062	3.7924	0.0004
Murder1973	0.7837	0.0424	18.4709	0.0000
I(Population/Area)	-1.7810	0.8362	-2.1299	0.0384

Some common residual plots pop out when you simply plot the model object.

```
plot(model.3)
```



If you're more into algorithmic variable selection, here's a stepwise method using AIC as the evaluation metric. I have used the "trace = 0" input to silence the step-by-step output, which details at which step variables were included and excluded during the process. I also told the function to step backward so that it removes unhelpful variables but never tries to add back variables.

```
model.4 <- step(lm(Murder1976 ~ Population + Income + Illiteracy +
                     LifeExp + HSGrad + Frost + Area + Murder1973 +
                     Assault + UrbanPop + Rape, data = joindata.1),
                direction = "backward", trace = 0)
kable(summary(model.4)$coef, digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	44.5600	16.7947	2.6532	0.0112
Population	0.0001	0.0000	2.6771	0.0105
Illiteracy	1.2133	0.4198	2.8904	0.0061
LifeExp	-0.5954	0.2369	-2.5135	0.0159
Murder1973	0.4480	0.1014	4.4179	0.0001
Assault	-0.0055	0.0038	-1.4272	0.1609
UrbanPop	-0.0268	0.0163	-1.6440	0.1076
Rape	0.1076	0.0279	3.8597	0.0004

Now that you have carpal tunnel from typing out all of those variable names, let's make that easier. Using "." in the model formula simply tells it to include all variables in the data frame that haven't yet been called. A "-" sign can then leave out variables you don't want.

```
model.4 <- step(lm(Murder1976 ~ . - State,
                   data = joindata.1), trace = 0)
kable(summary(model.4)$coef, digits = 4)
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	44.5600	16.7947	2.6532	0.0112
Population	0.0001	0.0000	2.6771	0.0105
Illiteracy	1.2133	0.4198	2.8904	0.0061
LifeExp	-0.5954	0.2369	-2.5135	0.0159
Murder1973	0.4480	0.1014	4.4179	0.0001
Assault	-0.0055	0.0038	-1.4272	0.1609
UrbanPop	-0.0268	0.0163	-1.6440	0.1076
Rape	0.1076	0.0279	3.8597	0.0004

This model seems to better than my first, having a higher adjusted R-squared. However, with so many variables on such a small dataset, we have run the risk of overfitting.

One last function we want to share shows us the Variance Inflation Factors for each predictor variable. Greater values indicate increased linear correlation between the predictor variables, and values too far above about 3.0 or 4.0 can lead to fitted coefficients with confusing values and high standard errors. With a thorough understanding of your data, you can work around or through such correlation issues, which are often referred to as "multicollinearity."

```
vif(model.4)
```

```
## Population Illiteracy    LifeExp Murder1973    Assault   UrbanPop       Rape
##   1.592092   2.508586   3.875046   7.477121   3.902666   2.141321   2.611877
```