In [35]:

```
import pandas as pd
import numpy as np
import statsmodels.api as sm
import sklearn.metrics as metrics
from sklearn.utils import resample
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Question 1.

Estimate the MSE of the model using the traditional approach. That is, fit the linear regression model using the entire dataset and calculate the mean squared error for the entire dataset. Present and discuss your results at a sinple, high level.

In [2]:

```
biden = pd.read_csv('nes2008.csv')
biden.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1807 entries, 0 to 1806
Data columns (total 6 columns):
biden     1807 non-null int64
female    1807 non-null int64
age       1807 non-null int64
educ      1807 non-null int64
dem       1807 non-null int64
rep       1807 non-null int64
dtypes: int64(6)
memory usage: 84.8 KB
```

In [6]:

```
x = biden[['female', 'age', 'educ', 'dem', 'rep']]
y = biden['biden']

x = sm.add_constant(x)
model1 = sm.OLS(y, x).fit()
model1.summary()
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[6]:

OLS Regression Results

Dep. Variable:	biden	**R-squared:**	0.282
Model:	OLS	**Adj. R-squared:**	0.280
Method:	Least Squares	**F-statistic:**	141.1
Date:	Mon, 03 Feb 2020	**Prob (F-statistic):**	1.50e-126
Time:	13:51:55	**Log-Likelihood:**	-7966.6
No. Observations:	1807	**AIC:**	1.595e+04
Df Residuals:	1801	**BIC:**	1.598e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	58.8113	3.124	18.823	0.000	52.683	64.939
female	4.1032	0.948	4.327	0.000	2.243	5.963
age	0.0483	0.028	1.708	0.088	-0.007	0.104
educ	-0.3453	0.195	-1.773	0.076	-0.727	0.037
dem	15.4243	1.068	14.442	0.000	13.330	17.519
rep	-15.8495	1.311	-12.086	0.000	-18.421	-13.278

Omnibus:	87.979	**Durbin-Watson:**	1.996
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	101.940
Skew:	-0.533	**Prob(JB):**	7.31e-23
Kurtosis:	3.466	**Cond. No.**	348.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [8]:

```
y_pred = model1.predict(x)
metrics.mean_squared_error(y, y_pred)
```

Out[8]:

```
395.2701692786484
```

The mean squared error for the entire dataset is 395.27. In the linear regression result, r-squared is 0.282, representing that this model only explains 28.2% variation in the dataset. For the features, the p-values of female, dem, and rep are pretty small. Therefore, it's fair to say that we have 99% confidence level to say that female and demographic have more active attitudes towards Joe Biden than male and republican. The coefficient of age is positive, education is negative. But the p-values are not that small.

Question 2.

In [12]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state = 0, test_si
```

In [13]:

```
x_train = sm.add_constant(x_train)
model2 = sm.OLS(y_train, x_train).fit()
model2.summary()
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[13]:

OLS Regression Results

Dep. Variable:	biden	**R-squared:**	0.290
Model:	OLS	**Adj. R-squared:**	0.286
Method:	Least Squares	**F-statistic:**	73.22
Date:	Mon, 03 Feb 2020	**Prob (F-statistic):**	2.42e-64
Time:	13:53:58	**Log-Likelihood:**	-3998.3
No. Observations:	903	**AIC:**	8009.
Df Residuals:	897	**BIC:**	8037.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	61.1071	4.479	13.642	0.000	52.316	69.899
female	1.6538	1.368	1.209	0.227	-1.032	4.339
age	0.0362	0.041	0.882	0.378	-0.044	0.117
educ	-0.3524	0.278	-1.268	0.205	-0.898	0.193
dem	15.7318	1.541	10.207	0.000	12.707	18.757
rep	-17.8384	1.896	-9.407	0.000	-21.560	-14.117

Omnibus:	21.887	**Durbin-Watson:**	2.026
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	22.939
Skew:	-0.389	**Prob(JB):**	1.04e-05
Kurtosis:	3.069	**Cond. No.**	346.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [25]:

```
x_test = sm.add_constant(x_test)
y_pred = model2.predict(x_test)
metrics.mean_squared_error(y_test, y_pred)
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[25]:

```
384.3787194770427
```

The MSE of question 2 is 384.38, which is different from what we calculate in question 1, 395.27. The dataset that we use to calculate MSE in question 1 is exactly what we use to train the model. However, in question 2, the dataset that we use to calculate MSE is the model has never seen before. Therefore, it is fair that the MSE in question 2 is smaller than question 1. Also, it's possible that the MSE in question 2 is bigger/ equal that question 1, because this depends on how we split the training set and the test set.

Question 3

In [22]:

```
mse1000 = []
for i in range(1000):
    xtra, xte, ytra, yte = train_test_split(x, y, random_state=i, test_size=0.5)
    xtra = sm.add_constant(xtra)
    model = sm.OLS(ytra, xtra).fit()
    #mlresult = mlmodel.fit()
    xte = sm.add_constant(xte)
    y_pred = model.predict(xte)
    mse1000.append(metrics.mean_squared_error(yte, y_pred))
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
```

In [31]:

```
sum(mse1000) / len(mse1000)
```

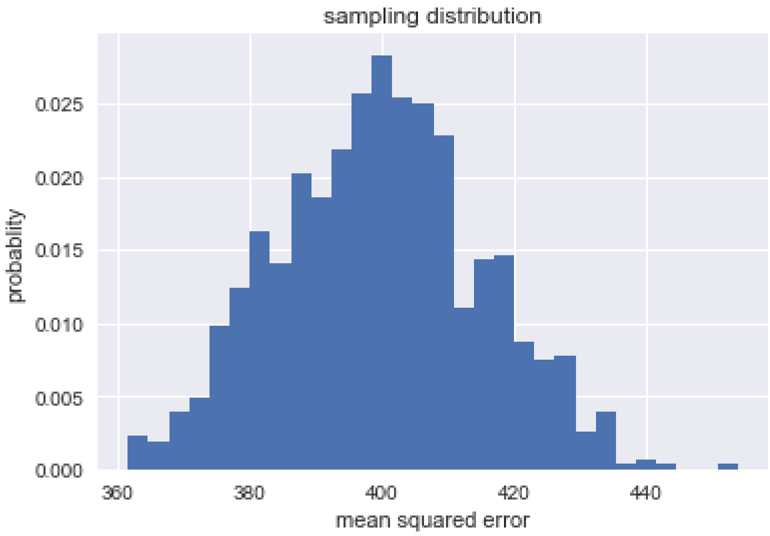
Out[31]:

```
399.6615061066417
```

In [33]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.hist(mse1000, normed=True, bins=30)
plt.xlabel('mean squared error')
plt.ylabel('probablity')
plt.title('sampling distribution')
plt.show()
```

```
/opt/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:3: Ma
tplotlibDeprecationWarning:
The 'normed' kwarg was deprecated in Matplotlib 2.1 and will be remove
d in 3.1. Use 'density' instead.
  This is separate from the ipykernel package so we can avoid doing im
ports until
```



The picture looks like a normal distribution, corresponding to the central limit theorem. The mean of MSE is 399.66. Recall that we get 395.27 in question 1 and 384.38 in question 2, 395.27 is closer to 384.38. This makes sense because 395.27 is to fit the entire dataset. And this is a little different from 399.6 because using bootstrap method helps to get close to the population mean.

Question 4

In [40]:

```
boot = resample(biden, replace=True, n_samples=1000, random_state=0)
boot.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1000 entries, 684 to 1168
Data columns (total 6 columns):
biden     1000 non-null int64
female    1000 non-null int64
age       1000 non-null int64
educ      1000 non-null int64
dem       1000 non-null int64
rep       1000 non-null int64
dtypes: int64(6)
memory usage: 54.7 KB
```

In [41]:

```
x_train = boot[['female', 'age', 'educ', 'dem', 'rep']]
y_train = boot[['biden']]
x_train = sm.add_constant(x_train)
model4 = sm.OLS(y_train, x_train).fit()
model4.summary()
```

```
/opt/anaconda3/lib/python3.7/site-packages/numpy/core/fromnumeric.py:2
495: FutureWarning: Method .ptp is deprecated and will be removed in a
future version. Use numpy.ptp instead.
  return ptp(axis=axis, out=out, **kwargs)
```

Out[41]:

OLS Regression Results

Dep. Variable:	biden	**R-squared:**	0.291
Model:	OLS	**Adj. R-squared:**	0.288
Method:	Least Squares	**F-statistic:**	81.71
Date:	Mon, 03 Feb 2020	**Prob (F-statistic):**	6.35e-72
Time:	14:10:21	**Log-Likelihood:**	-4420.4
No. Observations:	1000	**AIC:**	8853.
Df Residuals:	994	**BIC:**	8882.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	59.5482	4.174	14.266	0.000	51.357	67.739
female	1.8492	1.287	1.437	0.151	-0.676	4.374
age	0.0137	0.038	0.357	0.721	-0.062	0.089
educ	-0.3431	0.261	-1.315	0.189	-0.855	0.169
dem	18.3129	1.448	12.647	0.000	15.471	21.154
rep	-14.3282	1.785	-8.027	0.000	-17.831	-10.825

Omnibus:	31.600	**Durbin-Watson:**	2.099
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	33.771
Skew:	-0.441	**Prob(JB):**	4.64e-08
Kurtosis:	3.179	**Cond. No.**	345.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [44]:

```
y_pred = model4.predict(x_train)
metrics.mean_squared_error(y_train, y_pred)
```

Out[44]:

```
404.57987983796636
```

The numeric output comparsion of question 1 and question 4:

In [45]:

```
model1.summary()
```

Out[45]:

OLS Regression Results

Dep. Variable:	biden	**R-squared:**	0.282
Model:	OLS	**Adj. R-squared:**	0.280
Method:	Least Squares	**F-statistic:**	141.1
Date:	Mon, 03 Feb 2020	**Prob (F-statistic):**	1.50e-126
Time:	14:11:51	**Log-Likelihood:**	-7966.6
No. Observations:	1807	**AIC:**	1.595e+04
Df Residuals:	1801	**BIC:**	1.598e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	58.8113	3.124	18.823	0.000	52.683	64.939
female	4.1032	0.948	4.327	0.000	2.243	5.963
age	0.0483	0.028	1.708	0.088	-0.007	0.104
educ	-0.3453	0.195	-1.773	0.076	-0.727	0.037
dem	15.4243	1.068	14.442	0.000	13.330	17.519
rep	-15.8495	1.311	-12.086	0.000	-18.421	-13.278

Omnibus:	87.979	**Durbin-Watson:**	1.996
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	101.940
Skew:	-0.533	**Prob(JB):**	7.31e-23
Kurtosis:	3.466	**Cond. No.**	348.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [46]:

```
model4.summary()
```

Out[46]:

OLS Regression Results

Dep. Variable:	biden	**R-squared:**	0.291
Model:	OLS	**Adj. R-squared:**	0.288
Method:	Least Squares	**F-statistic:**	81.71
Date:	Mon, 03 Feb 2020	**Prob (F-statistic):**	6.35e-72
Time:	14:11:57	**Log-Likelihood:**	-4420.4
No. Observations:	1000	**AIC:**	8853.
Df Residuals:	994	**BIC:**	8882.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	59.5482	4.174	14.266	0.000	51.357	67.739
female	1.8492	1.287	1.437	0.151	-0.676	4.374
age	0.0137	0.038	0.357	0.721	-0.062	0.089
educ	-0.3431	0.261	-1.315	0.189	-0.855	0.169
dem	18.3129	1.448	12.647	0.000	15.471	21.154
rep	-14.3282	1.785	-8.027	0.000	-17.831	-10.825

Omnibus:	31.600	**Durbin-Watson:**	2.099
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	33.771
Skew:	-0.441	**Prob(JB):**	4.64e-08
Kurtosis:	3.179	**Cond. No.**	345.

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In general, the numerical outputs of question 1 and question 4 are pretty similiar. The R-squared in question 1 is 0.282, and 0.291 in question 4. The difference in coefficients are the bootstrap method has smaller estimates for female and age, and bigger estimates for education, demographic, and repbulican. A little bit bigger in question 4, but not very much. The p-value of female is 0.151 in question 4, comparing to 0.000 in question 1. The p-value of age gets bigger, you cannot say anything about a 0.721 p-value, and the p-value of education also beceomes bigger.
Generally, it's better to use bootstrap method because it's possible that the dataset cannot meet the distribution assumption, so the bootstrap method gives us a more accurate and robust result.

In [ ]:

```

```