PS03

April 23, 2023

1 PS03

1.1 Name: Xinyu Chang

[85]:
```
# import the packages
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from math import comb
```

1.1.1 1 Binomial distribution

Q1. Compute the expected value of this RV EX

[86]:
```
n = 3
p = 0.6
expected_value = p * n
print(expected_value)
```

```
1.7999999999999998
```

The expected value of this RV E X is exactly 1.8.

Q2. Compute the probability your get 0, 1, 2 or 3 "successes" I am using binomial cofficient functions comb in this problem.

[87]:
```
prob_0 = comb(n, 0) * p**0 * (1-p)**(n-0)
prob_1 = comb(n, 1) * p**1 * (1-p)**(n-1)
prob_2 = comb(n, 2) * p**2 * (1-p)**(n-2)
prob_3 = comb(n, 3) * p**3 * (1-p)**(n-3)
print("Probablity of getting 0 success is:", prob_0)
print("Probablity of getting 1 success is:", prob_1)
print("Probablity of getting 2 successes is:", prob_2)
print("Probablity of getting 3 successes is:", prob_3)
```

```
Probablity of getting 0 success is: 0.06400000000000002
Probablity of getting 1 success is: 0.28800000000000003
Probablity of getting 2 successes is: 0.43200000000000005
Probablity of getting 3 successes is: 0.21599999999999997
```

So the probabilities of getting 0, 1, 2, or 3 successes are 0.064, 0.288, 0.432, and 0.216, respectively.

Q3. Compute the variance of this RV Var X The formula of the variance of the RV Var X can be concluded as Var(X) = E[X^2] - (E[X])^2.

[88]:
```
variance = (0.064 * 0**2 + 0.288 * 1**2 + 0.432 * 2**2 + 0.216 * 3**2) - (1.8
↪** 2)
variance
```

[88]:
```
0.7199999999999998
```

The variance of this RV Var X is about 0.72.

Q4. Pick a sample size R (1000...100,000 are good numbers) and create a sample of size R of such random values.

[89]:
```
R = 1000
sample = np.random.binomial(3, 0.6, size=R)
sample[:10]  # just for check
```

[89]:
```
array([2, 2, 1, 0, 2, 2, 2, 1, 2, 2])
```

Q5. Compute the mean of your sample.

[90]:
```
mean_sample = np.mean(sample)
mean_sample
```

[90]:
```
1.828
```

The mean of the sample is 1.828 which is close to the EX value.

Q6. Compute the variance of your sample.

[92]:
```
var_sample = np.var(sample)
var_sample
```

[92]:
```
0.720416
```

The variance of the sample is 0.729 which is close to the Var X value.

Q7. Compute the expected number of cases with 0, 1, 2, and 3 successes for the sample size.

[93]:
```
success_0_EX = R * prob_0
success_1_EX = R * prob_1
success_2_EX = R * prob_2
success_3_EX = R * prob_3
print("The expected number of cases with 0 success:", success_0_EX)
print("The expected number of cases with 1 success:", success_1_EX)
print("The expected number of cases with 2 success:", success_2_EX)
```

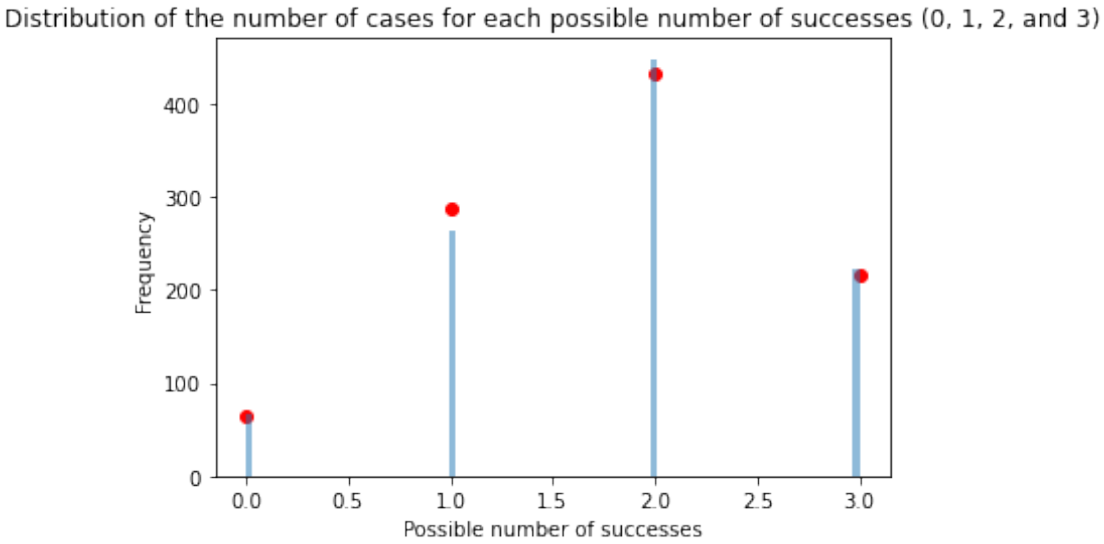
```
print("The expected number of cases with 3 success:", success_3_EX)
```

```
The expected number of cases with 0 success: 64.00000000000001
The expected number of cases with 1 success: 288.00000000000006
The expected number of cases with 2 success: 432.00000000000006
The expected number of cases with 3 success: 215.99999999999997
```

The expected number of cases with the sample size 1000 of 0, 1, 2, and 3 successes are 64, 288, 432, 216 respectively.

Q8. Make a histogram of your sample.

[94]:
```
successes = [0, 1, 2, 3]
counts = [success_0_EX, success_1_EX, success_2_EX, success_3_EX]
plt.hist(sample, alpha = 0.5, bins=100)
plt.scatter(successes, counts, c="red")
plt.title("Distribution of the number of cases for each possible number of␣
 ↪successes (0, 1, 2, and 3)")
plt.xlabel("Possible number of successes")
plt.ylabel("Frequency")
plt.show()
```



The resulting histogram shows the distribution of the sample and the expected number of cases for each possible number of successes (0, 1, 2, and 3). The histogram bars are centered at the possible values of X (0, 1, 2, and 3) and the heights of the bars represent the frequency of the sample that falls in each category. The red dots are placed closed to each bar and represent the expected proportion of the sample that falls in each category. The closer the expected numbers are to the actual frequencies in the sample, the better the match. Thus, the histogram matches the expected numbers fairly well.

1.1.2 2 Log-normal distribution

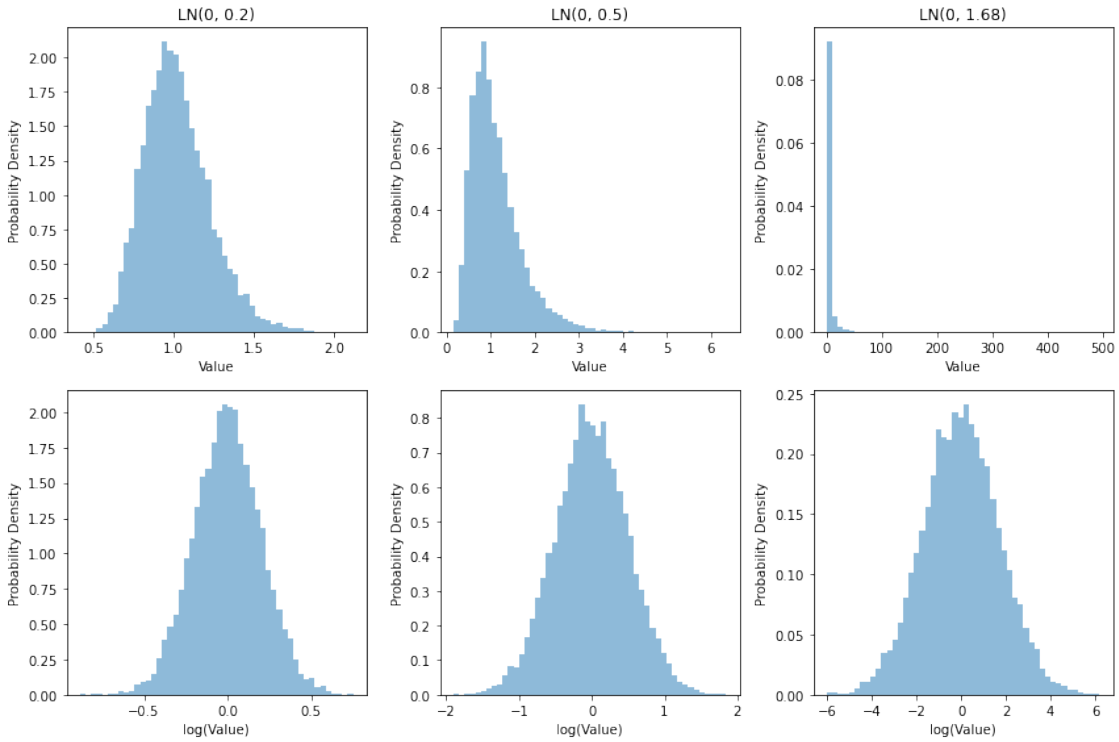
Q1. Choose your sample size S.

[115]:
```
S = 10000
```

Q2. Explore the shape of the distribution.

[116]:
```
sample_1 = np.random.lognormal(0, 0.2, S)
sample_2 = np.random.lognormal(0, 0.5, S)
sample_3 = np.random.lognormal(0, 1.68, S)
```

[117]:
```
fig, axs = plt.subplots(2, 3, figsize=(12, 8))

# Plot LN(0, 0.2)
mu, sigma = 0, 0.2
axs[0, 0].hist(sample_1, bins=50, density=True, alpha=0.5)
axs[0, 0].set_title(f"LN({mu}, {sigma})")
axs[0, 0].set_xlabel("Value")
axs[0, 0].set_ylabel("Probability Density")

axs[1, 0].hist(np.log(sample_1), bins=50, density=True, alpha=0.5)
axs[1, 0].set_xlabel("log(Value)")
axs[1, 0].set_ylabel("Probability Density")

# Plot LN(0, 0.5)
mu, sigma = 0, 0.5
axs[0, 1].hist(sample_2, bins=50, density=True, alpha=0.5)
axs[0, 1].set_title(f"LN({mu}, {sigma})")
axs[0, 1].set_xlabel("Value")
axs[0, 1].set_ylabel("Probability Density")

axs[1, 1].hist(np.log(sample_2), bins=50, density=True, alpha=0.5)
axs[1, 1].set_xlabel("log(Value)")
axs[1, 1].set_ylabel("Probability Density")

# Plot LN(0, 1.68)
mu, sigma = 0, 1.68
axs[0, 2].hist(sample_3, bins=50, density=True, alpha=0.5)
axs[0, 2].set_title(f"LN({mu}, {sigma})")
axs[0, 2].set_xlabel("Value")
axs[0, 2].set_ylabel("Probability Density")

axs[1, 2].hist(np.log(sample_3), bins=50, density=True, alpha=0.5)
axs[1, 2].set_xlabel("log(Value)")
axs[1, 2].set_ylabel("Probability Density")

plt.tight_layout()
plt.show()
```



From the linear-linear scale histograms, we can see that all three distributions are right skewed, with longer tails towards higher values. The higher of the sigma value, the more skewed of the distribution. From the log-linear scale histograms, the shapes of the distributions are almost normal distribution.

Q3. Look at the histograms and tell-what do you think, which one describes the least unequal distribution, and which one the most unequal distribution? From the linear-linear scale histograms, we can find that the sample_3(mu_3, sigma_3 = 0, 1.68) has the most unequal distribution and the sample_1 (mu_1, sigma_1 = 0, 0.2) has the least unequal distribution. The parameter determines the spread, with smaller giving a more equal distribution and larger giving a more unequal one. It can be observed by graph that the sample_3 has a largest standard deviation and the sample_1 has a smallest standard deviation.

Q4. Compute sample means and variance

[118]:
```
sample_mean_1 = np.mean(sample_1)
theoretical_mean_1 = np.exp(0 + 0.5 * 0.2**2)
sample_variance_1 = np.var(sample_1)
theoretical_variance_1 = np.exp(2*0 + 0.2**2) * (np.exp(0.2**2) - 1)
print("sample_mean_1:",sample_mean_1)
print("theoretical_mean_1:",theoretical_mean_1)
print("sample_variance_1:",sample_variance_1)
print("theoretical_variance_1:",theoretical_variance_1)
```

```
sample_mean_1: 1.0202853146884183
theoretical_mean_1: 1.0202013400267558
sample_variance_1: 0.043165713828662966
theoretical_variance_1: 0.04247629348257031
```

[119]:
```
sample_mean_2 = np.mean(sample_2)
theoretical_mean_2 = np.exp(0 + 0.5 * 0.5**2)
sample_variance_2 = np.var(sample_2)
theoretical_variance_2 = np.exp(2*0 + 0.5**2) * (np.exp(0.5**2) - 1)
print("sample_mean_2:",sample_mean_2)
print("theoretical_mean_2:",theoretical_mean_2)
print("sample_variance_2:",sample_variance_2)
print("theoretical_variance_2:",theoretical_variance_2)
```

```
sample_mean_2: 1.1294520244613393
theoretical_mean_2: 1.1331484530668263
sample_variance_2: 0.36012433682343986
theoretical_variance_2: 0.3646958540123865
```

[120]:
```
sample_mean_3 = np.mean(sample_3)
theoretical_mean_3 = np.exp(0 + 0.5 * 1.68**2)
sample_variance_3 = np.var(sample_3)
theoretical_variance_3 = np.exp(2*0 + 1.68**2) * (np.exp(1.68**2) - 1)
print("sample_mean_3:",sample_mean_3)
print("theoretical_mean_3:",theoretical_mean_3)
print("sample_variance_3:",sample_variance_3)
print("theoretical_variance_3:",theoretical_variance_3)
```

```
sample_mean_3: 4.314485036811766
theoretical_mean_3: 4.1008735008239405
sample_variance_3: 231.447823365115
theoretical_variance_3: 265.99982369887033
```

From the output, there are similar values between the sample means and variance and the theoretical mean and variance for the sample 1(mu_1, sigma_1 = 0, 0.2) and sample 2(mu_2, sigma_2 = 0, 0.5) except for the sample3(mu_3, sigma_3 = 0, 1.68). So, there is less inequality in the sample 1 and sample 2, because the theoretical and sample values are similar.However, for the sample 3, there are more inequality because there is some difference between the sample variance in sample 3 and the theoretical variance of sample 3.

Q5. Compute the Pareto ratio of all three samples.

[121]:
```
def pareto_ratio(data):
    sorted_data = sorted(data, reverse=True)
    total_sum = sum(sorted_data)
    for i in range(99, 50, -1):
        percentile = np.percentile(sorted_data, i)
        ratio = sum([x for x in sorted_data if x >= percentile]) / total_sum
        if ratio * 100 > i:
```

```
            return (100 - i, ratio)
    return None
```

[122]:
```
pareto_ratio(sample_1)
```

[122]: (47, 0.5498425305634205)

[123]:
```
pareto_ratio(sample_2)
```

[123]: (41, 0.6077202143966621)

[124]:
```
pareto_ratio(sample_3)
```

[124]: (20, 0.8089958523544851)

The most unequal distribution is the sample_3, where only 20% of the population accounts for 80.9% of the total effect. This closely follows the Pareto principle, suggesting a highly skewed distribution. The most equal distribution is the sample_1, where 47% of the population is responsible for 54.8% of the total effect. This distribution is closer to an equal distribution, as the percentages are not as skewed as in the other cases.The results correspond to what I did on the the histograms.

1.1.3 3 Inequality in data

Q1.Load both datasets and do basic checks—do the values of interest (number of citations and income) look reasonable?

[125]:
```
treatment_data = pd.read_csv('treatment.csv', sep = '\t')
treatment_data.head(10)
```

[125]:

	treat	age	educ	ethn	married	re74	re75	re78	u74	u75
0	True	37	11	black	True	0.0	0.0	9930.05	True	True
1	True	30	12	black	False	0.0	0.0	24909.50	True	True
2	True	27	11	black	False	0.0	0.0	7506.15	True	True
3	True	33	8	black	False	0.0	0.0	289.79	True	True
4	True	22	9	black	False	0.0	0.0	4056.49	True	True
5	True	23	12	black	False	0.0	0.0	0.00	True	True
6	True	32	11	black	False	0.0	0.0	8472.16	True	True
7	True	22	16	black	False	0.0	0.0	2164.02	True	True
8	True	19	9	black	False	0.0	0.0	8173.91	True	True
9	True	21	13	black	False	0.0	0.0	17094.60	True	True

[126]:
```
treatment_data.shape
```

[126]: (2675, 10)

There are 2675 rows and 10 columns in the treatment dataset. The sample shows that the re78 variable in treatment_data represents the real income of labor market program participants in 1978, and the first 10 lines of data look reasonable and the data fits the type that the column defines.

[127]:
```
citations_data = pd.read_csv('mag-30k-citations.csv', sep = '\t')
citations_data.head(10)
```

[127]:

	id	citations	year
0	1926704088	10	2004
1	1590836146	0	1971
2	2035484488	3	2007
3	1965723909	0	2009
4	2021155615	42	1998
5	2228695328	0	2005
6	2197938220	0	2002
7	2024089647	13	1949
8	1893307378	60	1996
9	141316683	0	2005

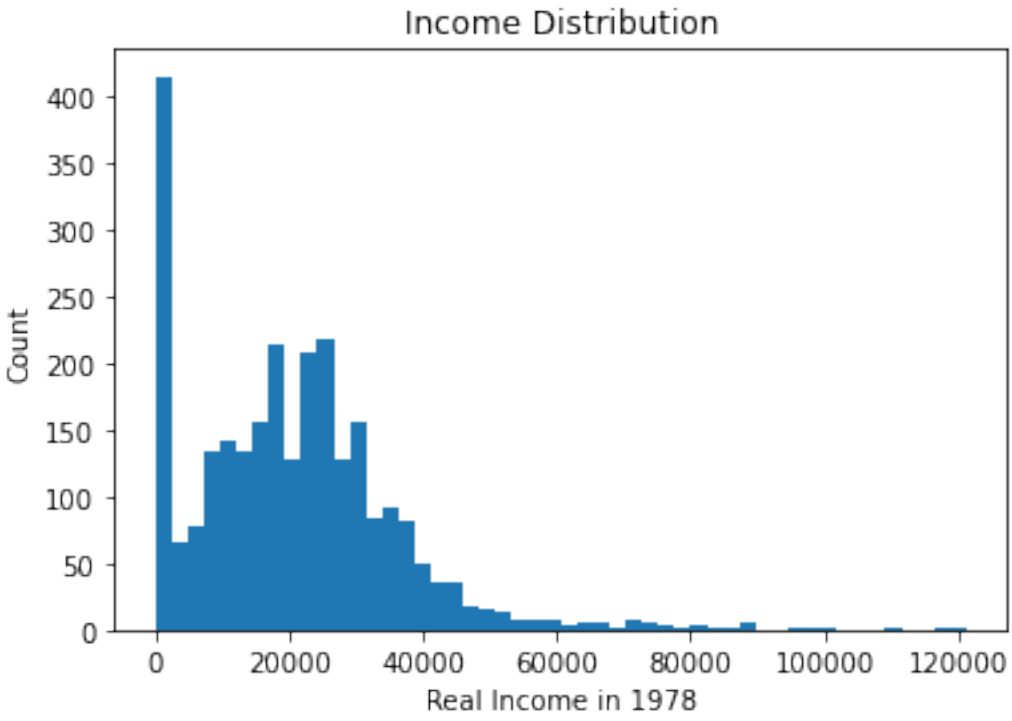
[128]:
```
citations_data.shape
```

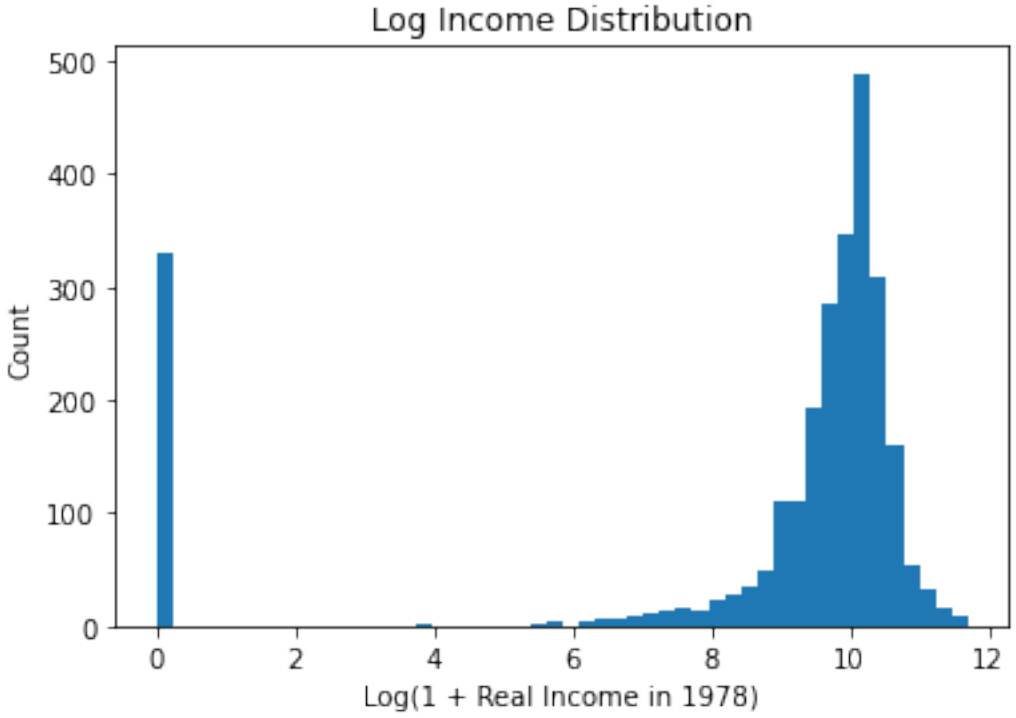
[128]: (30000, 3)

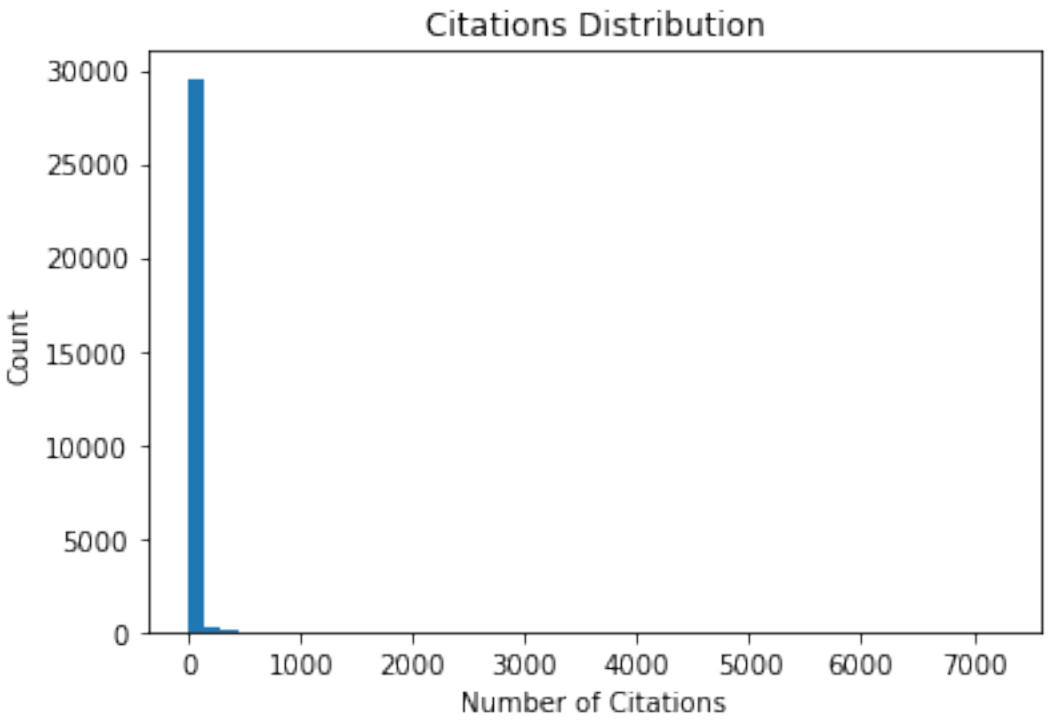
There are 30,000 rows and 3 columns in the mag-30k-citations dataset. The sample shows that the citations variable in citations_data represents the number of times research papers have been cited, and the first 10 lines of data look reasonable and the data fits the type that the column defines.

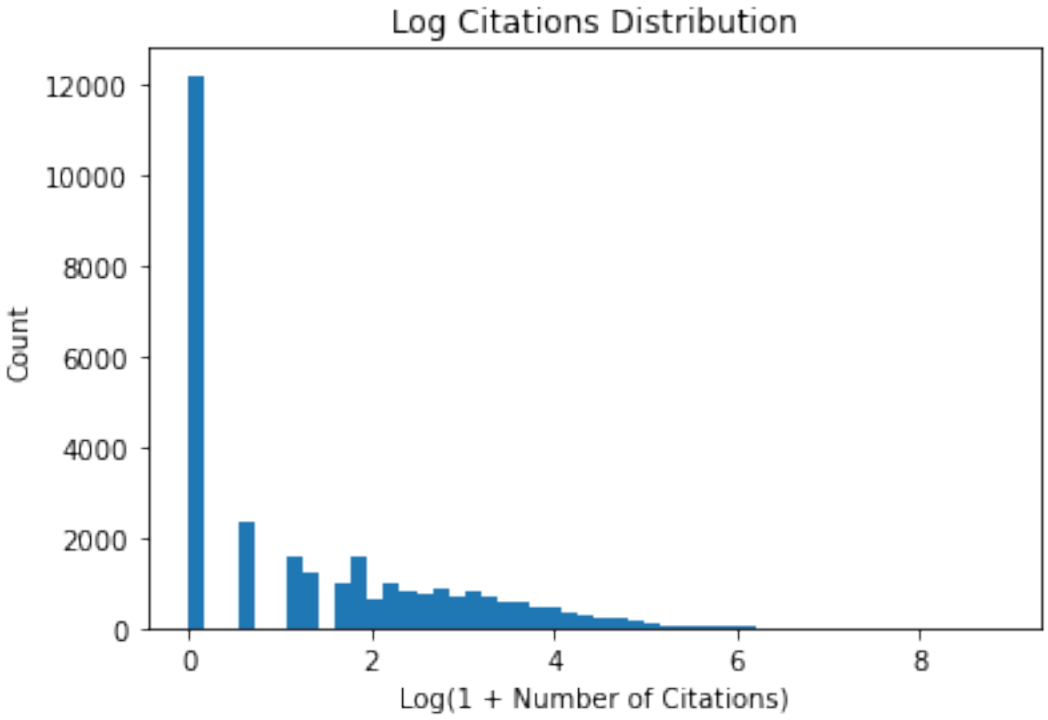
Q2. Show the distribution of income and citations on a histogram.

[129]:
```
plt.hist(treatment_data['re78'], bins=50)
plt.title('Income Distribution')
plt.xlabel('Real Income in 1978')
plt.ylabel('Count')
plt.show()

plt.hist(np.log(1+treatment_data['re78']), bins=50)
plt.title('Log Income Distribution')
plt.xlabel('Log(1 + Real Income in 1978)')
plt.ylabel('Count')
plt.show()

plt.hist(citations_data['citations'], bins=50)
plt.title('Citations Distribution')
plt.xlabel('Number of Citations')
plt.ylabel('Count')
plt.show()

plt.hist(np.log(1+citations_data['citations']), bins=50)
plt.title('Log Citations Distribution')
plt.xlabel('Log(1 + Number of Citations)')
plt.ylabel('Count')
plt.show()
```


Income Distribution
Count
0
50
100
150
200
250
300
350
400
0
20000
40000
60000
80000
100000
120000
Real Income in 1978


Log Income Distribution
Count
0
100
200
300
400
500
0
2
4
6
8
10
12
Log(1 + Real Income in 1978)


Citations Distribution
Count
30000
25000
20000
15000
10000
5000
0
0
1000
2000
3000
4000
5000
6000
7000
Number of Citations


Log Citations Distribution
Count
12000
10000
8000
6000
4000
2000
0
0
2
4
6
8
Log(1 + Number of Citations)

The Income distribution graph and the Citation distribution histogram reveal that both the income and citation distributions are significantly right skewed. For the Log income distribtuion graph and the log citations distribution graph, the log income distribution graph appears to be more symmetric although the value of log income of 0 is high, the reason maybe caused by a high number of low income populaton or non income population.However, the log citations distribution is a pareto distribution that has a highly skewed distribution.

Q3. Compute sample mean and standard deviation for both datasets.

[130]:
```
income_mean = np.mean(treatment_data['re78'])
income_std = np.std(treatment_data['re78'])
print('Income Mean:', income_mean)
print('Income Standard Deviation:', income_std)
print()

citations_mean = np.mean(citations_data['citations'])
citations_std = np.std(citations_data['citations'])
print('Citations Mean:', citations_mean)
print('Citations Standard Deviation:', citations_std)
print()
```

```
Income Mean: 20502.37607865417
Income Standard Deviation: 15629.597534767552

Citations Mean: 15.605633333333333
Citations Standard Deviation: 79.1767834759606
```

The mean of income is 20502.38, and the standard deviation is 15629.60.
The mean of citations is 15.61, and the standard deviation is 79.18.
When comparing the mean and standard deviation of each dataset, the standard deviation is approximately 1.3 times smaller than the mean, indicating a moderate level of inequality in the dataset. However, for the citation dataset, the standard deviation is approximately 5.3 times larger than the mean, indicating a high level of inequality in the dataset.

Q4. Compute the Pareto ratio for both distributions.

[131]:
```
pareto_ratio(treatment_data['re78'])
```

[131]:
```
(37, 0.6474562618876838)
```

For the treatment data, this pareto ratio indicates that top 37% of the population is responsible for 64.7% of the total effect (i.e., total income). This indicates an unequal distribution but not as extreme as the 80/20 rule.

[132]:
```
pareto_ratio(citations_data['citations'])
```

[132]:
```
(17, 0.8325220166222027)
```

For the citations dataset, 17% of the papers are responsible for 83.3% of the total citations. This distribution is more unequal and closely follows the Pareto principle.

1.1.4 4 Variance of mean

Q1. Pick your number of repetitions R.

[133]:
```
R = 100000
```

Q2. Now do R times in a loop: create a single realization of this RV.

[134]:
```
S = 1
N = 3
p = 0.6
pair_means = np.empty(shape=R)
for i in range(R):
  x = np.random.binomial(N, p, size=S)
  pair_means[i] = np.mean(x)
```

Q3. Compute variance of your sample.

[135]:
```
var_sample = np.var(pair_means)
var_sample
```

[135]:
```
0.7172084375
```

The variance of my sample is 0.72.

Q4. Now do a loop of R again, but this time create a pair of realizations (S = 2).

[136]:
```
S = 2
pair_means_ = np.empty(shape=R)
for i in range(R):
  x = np.random.binomial(N, p, size=S)
  pair_means[i] = np.mean(x)
```

Q5. Compute variance of your pair means.

[137]:
```
var_pair_means = np.var(pair_means)
var_pair_means
```

[137]:
```
0.359855999375
```

The variance of the pair mean is 0.36.

Q6. Repeat with S = 10-tuples (each time create 10 realizations in the loop and compute mean) and with S = 100-tuples.

[138]:
```
S = 10
ten_pair_means = np.empty(shape=R)
for i in range(R):
```

```
  x = np.random.binomial(N, p, size=S)
  ten_pair_means[i] = np.mean(x)

S = 100
hundred_pair_means = np.empty(shape=R)
for i in range(R):
  x = np.random.binomial(N, p, size=S)
  hundred_pair_means[i] = np.mean(x)

var_ten_tuple_means = np.var(ten_pair_means)
var_hundred_tuple_means = np.var(hundred_pair_means)
```

[139]:
```
var_ten_tuple_means
```

[139]:
```
0.07137869502399999
```

The variance with S = 10 tuples is 0.071.

[141]:
```
var_hundred_tuple_means
```

[141]:
```
0.0072020443640000015
```

The variance with S = 100 tuples is 0.0072.

Q7. Explain why variance of mean gets smaller when sample size S gets larger. One method to consider it is through the concept of "precision" in estimating the actual population mean.The sample mean is an estimate of the true population mean when a population is sampled. Due to the fact that our sample is a subset of the population, there will be some uncertainty or "error" in our estimate of the population mean.This error is quantified by the variance of the sample mean, which quantifies the amount by which sample means from various samples would vary if we repeatedly drew new samples from the same population.As the sample size S increases, each individual observation has a diminishing effect on the sample mean. In other words, the mean of the sample becomes more stable and less susceptible to random fluctuations in the data. This implies that the variance of the sample mean decreases as sample size increases, as the sources of variability in the sample mean diminish.

I spent almost 8 hours in this PS.