Runtimes

November 18, 2021

We will analyze two datasets to evaluate the empirical behavior of the algorithm: one in which the divisor size m is held constant and the dividend size n varies, and the other in which the dividend size n is held constant and the divisor size m varies.

Assuming arithmetic operations have complexity $\mathcal{O}(m)$, we expect to see behavior approximately modeled by

$$F_1(n, m) = m(n - m) = nm - m^2$$

For constant m, we expect to observe a positive linear relationship between n and runtime:

$$F(n) \approx n$$

For constant n, we expect to observe an inverse quadratic relationship between m and runtime:

$$F(m) \approx -m^2 + m$$

[85]:
```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import skew, kurtosis, kurtosistest
import statsmodels.api as sm
import statsmodels.formula.api as smf
from scipy.stats import shapiro
%matplotlib inline
```

0.1 $F(n)$: Variable Dividend Size - $n = \{12, 13, ..., 10000\}, m = 10$

[86]:
```
dividend_df = pd.read_csv('dividend_log.txt', header=None)
dividend_df = pd.DataFrame(dividend_df[0].str.split().tolist()).astype(int)
dividend_df.columns = ['return_value', 'dividend_size', 'divisor_size',␣
↪'runtime']
dividend_df['len_diff'] = dividend_df['dividend_size'] -␣
↪dividend_df['divisor_size']
dividend_df
```

[86]:

```
      return_value  dividend_size  divisor_size  runtime  len_diff
0                1             12            10        0         2
1                1             12            10        0         2
2                1             13            10        0         3
3                1             15            10        0         5
4                1             16            10        0         6
…              …             …           …       …       …
9984             1           9995            10       18      9985
9985             1           9997            10       18      9987
9986             1           9998            10       18      9988
9987             1           9999            10       19      9989
9988             1          10000            10       18      9990

[9989 rows x 5 columns]
```

0.1.1 $F(n)$ Plot - Fit 1st Order Polynomial to Dataset - $F(n) \approx n$

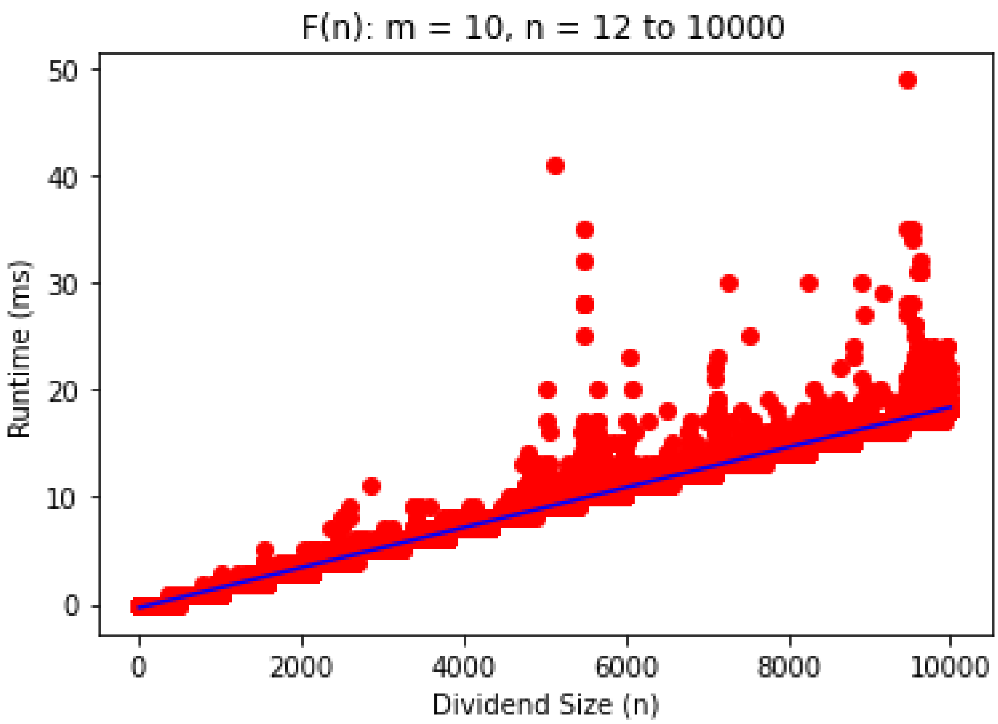
We observe a positive, approximately linear relationship with a fair amount of longer runtimes.

[87]:

```python
results = smf.ols(formula='runtime ~ len_diff', data=dividend_df).fit()
dividend_df['regression'] = results.fittedvalues
dividend_df['error'] = dividend_df['runtime'] - dividend_df['regression']
# print(shapiro(dividend_df['error']))
plt.scatter(dividend_df['len_diff'], dividend_df['runtime'], color='red')
plt.title('F(n): m = 10, n = 12 to 10000')
plt.xlabel('Dividend Size (n)')
plt.ylabel('Runtime (ms)')
plt.plot(dividend_df['dividend_size'], dividend_df['regression'], color='blue')
print(results.summary())
plt.savefig('dividend.png')
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                runtime   R-squared:                       0.956
Model:                            OLS   Adj. R-squared:                  0.956
Method:                 Least Squares   F-statistic:                 2.150e+05
Date:                Thu, 18 Nov 2021   Prob (F-statistic):               0.00
Time:                        09:29:31   Log-Likelihood:                -15699.
No. Observations:                9989   AIC:                         3.140e+04
Df Residuals:                    9987   BIC:                         3.142e+04
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.3513      0.023    -15.065      0.000      -0.397      -0.306
len_diff       0.0019   4.04e-06    463.724      0.000       0.002       0.002
```

```
==============================================================================
Omnibus:                    16689.004   Durbin-Watson:                   1.249
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         17734449.205
Skew:                          11.236   Prob(JB):                         0.00
Kurtosis:                     208.194   Cond. No.                     1.15e+04
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.15e+04. This might indicate that there are
strong multicollinearity or other numerical problems.
```

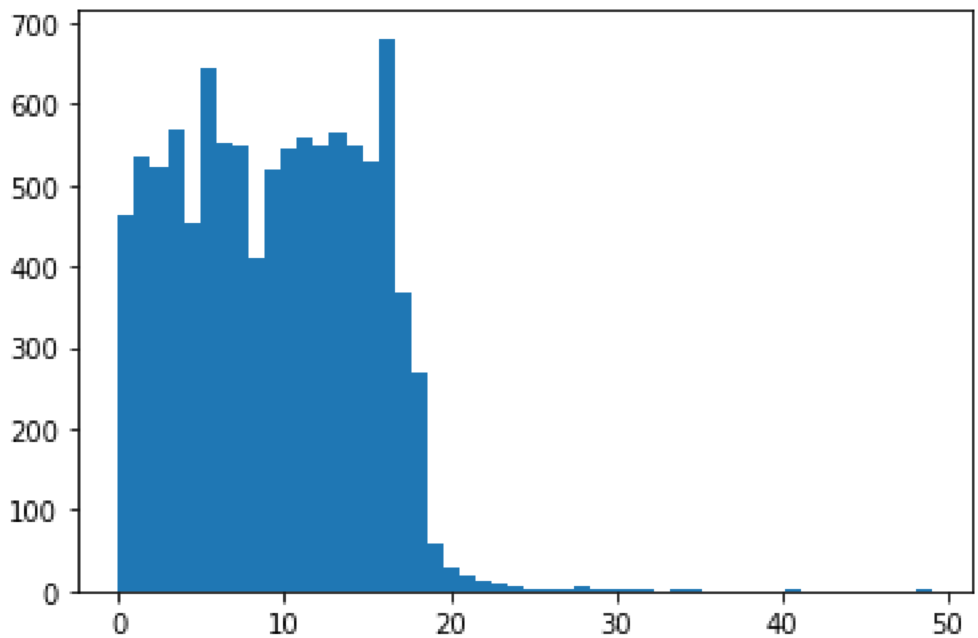


0.1.2 $F(n)$ Runtime Histogram

[88]:
```python
plt.hist(dividend_df['runtime'], bins=50)
```

[88]:
```
(array([464., 535., 522., 569., 452., 644., 553., 548., 412., 520., 544.,
        559., 550., 564., 549., 529., 681., 367., 268.,  58.,  29.,  19.,
         11.,   9.,   6.,   3.,   2.,   2.,   5.,   1.,   3.,   3.,   2.,
          0.,   1.,   3.,   0.,   0.,   0.,   0.,   0.,   1.,   0.,   0.,
          0.,   0.,   0.,   0.,   0.,   1.]),
```

```
array([ 0.  ,  0.98,  1.96,  2.94,  3.92,  4.9 ,  5.88,  6.86,  7.84,
        8.82,  9.8 , 10.78, 11.76, 12.74, 13.72, 14.7 , 15.68, 16.66,
       17.64, 18.62, 19.6 , 20.58, 21.56, 22.54, 23.52, 24.5 , 25.48,
       26.46, 27.44, 28.42, 29.4 , 30.38, 31.36, 32.34, 33.32, 34.3 ,
       35.28, 36.26, 37.24, 38.22, 39.2 , 40.18, 41.16, 42.14, 43.12,
       44.1 , 45.08, 46.06, 47.04, 48.02, 49.  ]),
<a list of 50 Patch objects>)
```



0.1.3 $F(n)$ Runtime Skew & Kurtosis

[89]:
```
print(f'Runtime skew = {skew(dividend_df["runtime"])}')
print(f'Runtime kurtosis {kurtosis(dividend_df["runtime"], fisher=False)}')
```

```
Runtime skew = 0.1981395618499291
Runtime kurtosis 2.654086484509155
```

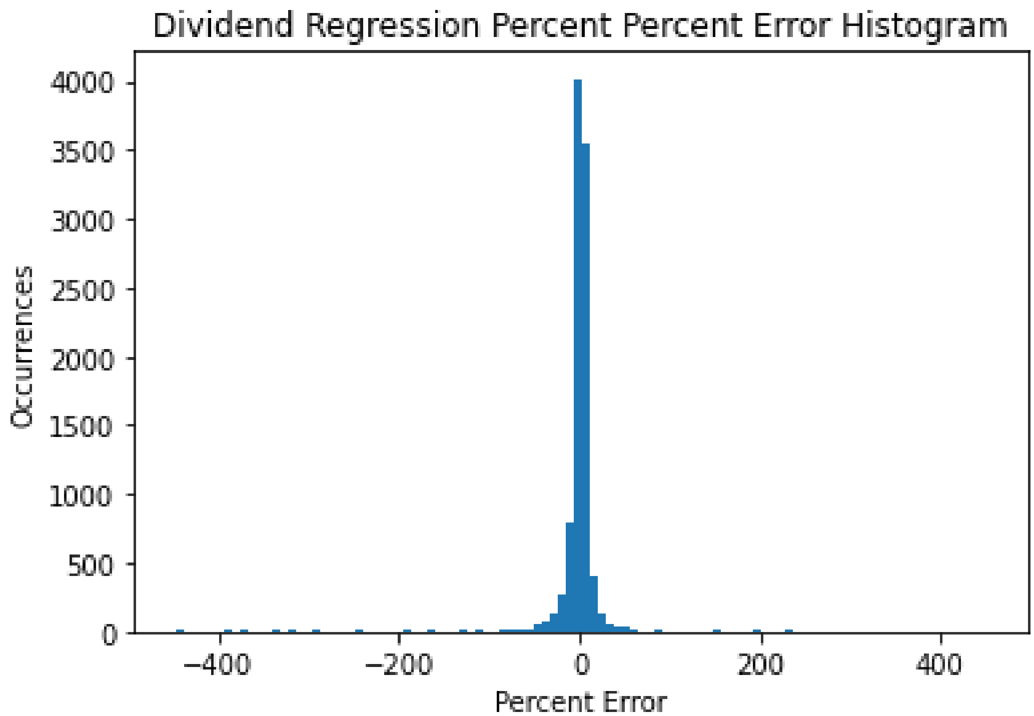
0.1.4 $F(n)$ Regression t-Values

[90]:
```
print(results.tvalues)
```

```
Intercept    -15.064792
len_diff     463.723667
dtype: float64
```

0.1.5 $F(n)$ Regression Percent Error Distribution

[91]:
```
dividend_df['runtime'] = dividend_df['runtime'].replace(0, .1)  # Avoid␣
 ↪division by 0
dividend_df['pct_err'] = 100*(dividend_df['regression'] -␣
 ↪dividend_df['runtime'])/dividend_df['runtime']
hist = np.histogram(dividend_df['pct_err'], 100)
plt.hist(dividend_df['pct_err'], bins=100)
plt.title('Dividend Regression Percent Percent Error Histogram')
plt.xlabel('Percent Error')
plt.ylabel('Occurrences')
plt.savefig('dividend_hist.png')
```



0.1.6 $F(n)$ Regression Percent Error Sample Statistical Moments

Sample moments indicate high peak with large tails, and slight negative skew - a tendency to underestimate runtimes.

[92]:
```
print('1st ORDER REGRESSION PERCENT ERROR:')
print(f'Mean = {round(np.mean(dividend_df["pct_err"]), 3)}')
print(f'Variance = {round(np.var(dividend_df["pct_err"]), 3)}')
print(f'Skew = {round(skew(dividend_df["pct_err"]), 3)}')
```

```
print(f'Kurtosis = {round(kurtosis(dividend_df["pct_err"]), 3)}')
```

```
1st ORDER REGRESSION PERCENT ERROR:
Mean = 0.115
Variance = 3121.917
Skew = -0.498
Kurtosis = 33.683
```

[93]:
```
dividend_df['pct_err'].describe()
```

[93]:
```
count    9989.000000
mean        0.114648
std        55.876918
min      -447.558716
25%        -2.854447
50%         1.594036
75%         5.373618
max       452.320859
Name: pct_err, dtype: float64
```

0.2 $F(m)$: Variable Divisor Size - $m = \{2, 3, ..., 9999\}, n = 10000$

[94]:
```
divisor_df = pd.read_csv('divisor_log.txt', header=None)
divisor_df = pd.DataFrame(divisor_df[0].str.split().tolist()).astype(int)
divisor_df.columns = ['return_value', 'dividend_size', 'divisor_size','runtime']
divisor_df
```

[94]:

	return_value	dividend_size	divisor_size	runtime
0	1	10000	2	6
1	1	10000	3	7
2	1	10000	4	9
3	1	10000	5	10
4	1	10000	6	12
…	…	…	…	…
9993	1	10000	9995	92
9994	1	10000	9996	92
9995	1	10000	9997	92
9996	1	10000	9998	89
9997	1	10000	9999	89

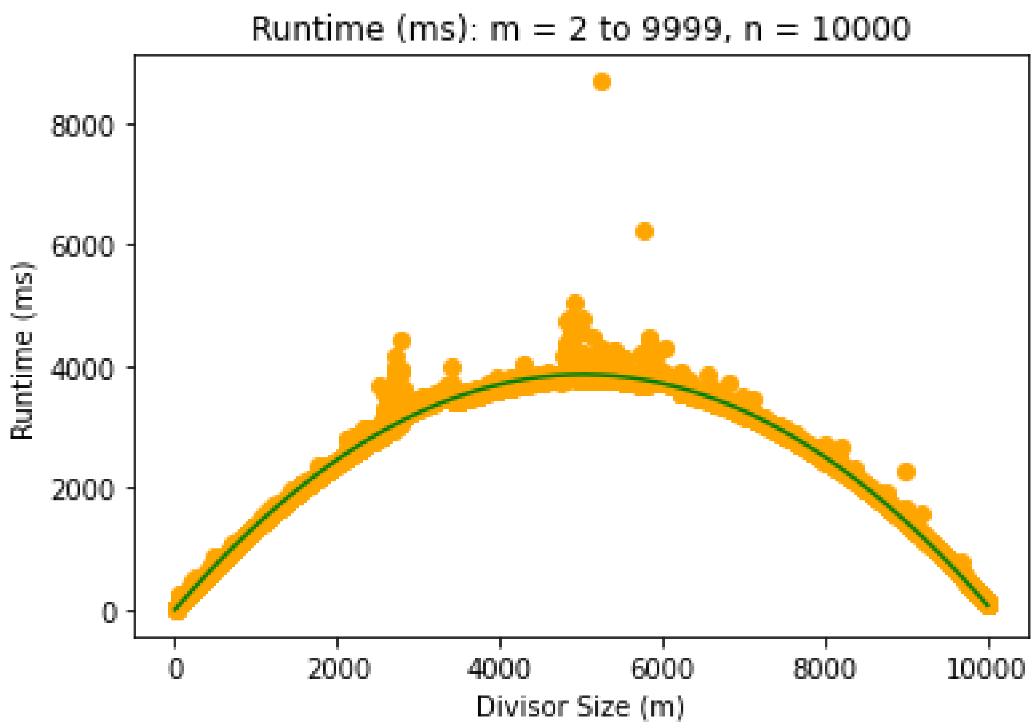
[9998 rows x 4 columns]

0.2.1 $F(m)$ Plot: Fit 2nd Order Polynomial to Dataset - $F(m) \approx -m^2 + m$

We observe a negative quadratic relationship with a trace amount of extreme runtimes.

[95]:
```
divisor_df['divisor_size_square'] = divisor_df['divisor_size']**2
results = smf.ols(formula='runtime ~ divisor_size + divisor_size_square',␣
 ↪data=divisor_df).fit()
divisor_df['regression'] = results.fittedvalues
plt.scatter(divisor_df['divisor_size'], divisor_df['runtime'], color='orange')
plt.plot(divisor_df['divisor_size'], divisor_df['regression'], color='green')
plt.title('Runtime (ms): m = 2 to 9999, n = 10000')
plt.xlabel('Divisor Size (m)')
plt.ylabel('Runtime (ms)')
print(results.summary())
plt.savefig('divisor.png')
```

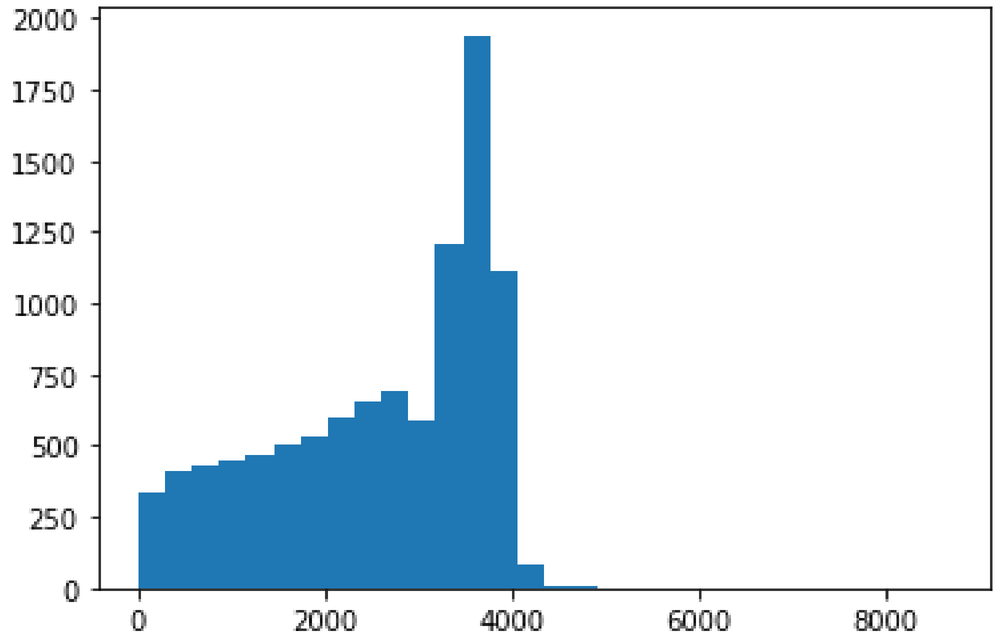
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                runtime   R-squared:                       0.993
Model:                            OLS   Adj. R-squared:                  0.993
Method:                 Least Squares   F-statistic:                 7.029e+05
Date:                Thu, 18 Nov 2021   Prob (F-statistic):               0.00
Time:                        09:29:32   Log-Likelihood:                -59863.
No. Observations:                9998   AIC:                         1.197e+05
Df Residuals:                    9995   BIC:                         1.198e+05
Df Model:                           2
Covariance Type:            nonrobust
================================================================================
=======
                          coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-------
Intercept             -11.2542      2.895     -3.888      0.000     -16.928
-5.580
divisor_size            1.5413      0.001   1153.020      0.000       1.539
1.544
divisor_size_square    -0.0002   1.29e-07  -1185.474      0.000      -0.000
-0.000
==============================================================================
Omnibus:                    20503.866   Durbin-Watson:                   0.991
Prob(Omnibus):                  0.000   Jarque-Bera (JB):        198180670.199
Skew:                          16.702   Prob(JB):                         0.00
Kurtosis:                     691.922   Cond. No.                     1.34e+08
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.34e+08. This might indicate that there are
strong multicollinearity or other numerical problems.
```



0.2.2 $F(m)$ Runtime Histogram

[96]:
```
plt.hist(divisor_df['runtime'], bins=30)
```

[96]:
```
(array([3.320e+02, 4.080e+02, 4.320e+02, 4.470e+02, 4.680e+02, 5.020e+02,
        5.290e+02, 5.990e+02, 6.510e+02, 6.890e+02, 5.840e+02, 1.206e+03,
        1.940e+03, 1.117e+03, 7.800e+01, 1.000e+01, 3.000e+00, 1.000e+00,
        0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00, 0.000e+00, 0.000e+00,
        0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 0.000e+00, 1.000e+00]),
 array([6.00000000e+00, 2.95166667e+02, 5.84333333e+02, 8.73500000e+02,
        1.16266667e+03, 1.45183333e+03, 1.74100000e+03, 2.03016667e+03,
        2.31933333e+03, 2.60850000e+03, 2.89766667e+03, 3.18683333e+03,
        3.47600000e+03, 3.76516667e+03, 4.05433333e+03, 4.34350000e+03,
        4.63266667e+03, 4.92183333e+03, 5.21100000e+03, 5.50016667e+03,
        5.78933333e+03, 6.07850000e+03, 6.36766667e+03, 6.65683333e+03,
        6.94600000e+03, 7.23516667e+03, 7.52433333e+03, 7.81350000e+03,
        8.10266667e+03, 8.39183333e+03, 8.68100000e+03]),
 <a list of 30 Patch objects>)
```



0.2.3 $F(m)$ Runtime Skew & Kurtosis

[97]:
```
print(f'Runtime skew = {skew(divisor_df["runtime"])}')
print(f'Runtime kurtosis {kurtosis(divisor_df["runtime"], fisher=False)}')
```

```
Runtime skew = -0.5906818055727012
Runtime kurtosis 2.1884923856929657
```

0.2.4 $F(m)$ Regression t-Values

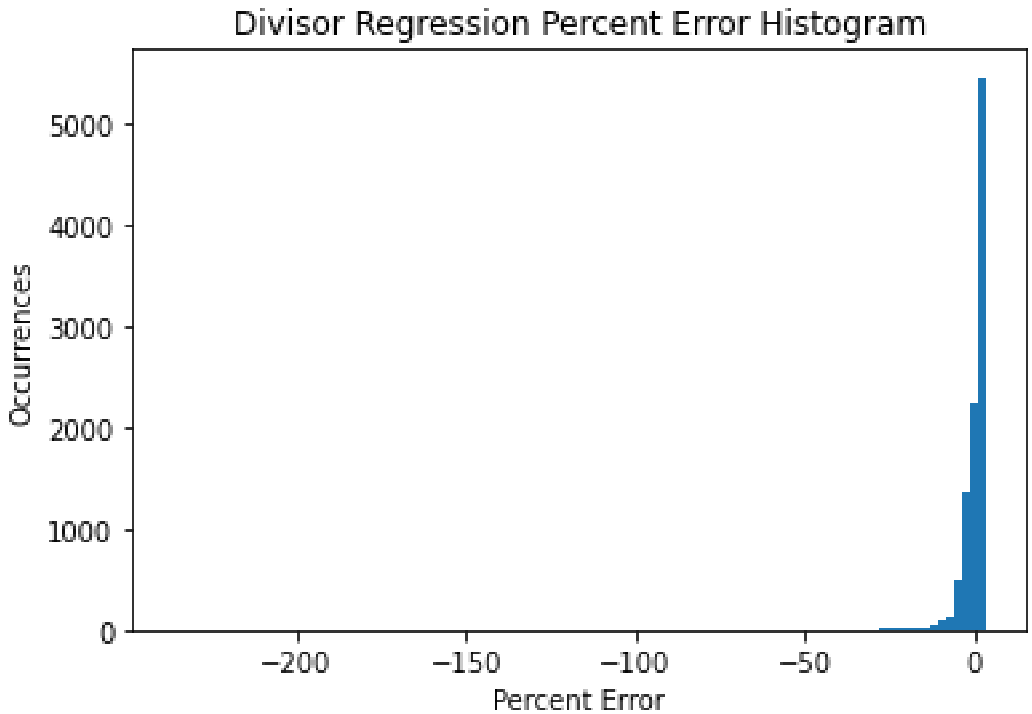
[98]:
```
print(results.tvalues)
```

```
Intercept                 -3.888035
divisor_size            1153.020022
divisor_size_square    -1185.473905
dtype: float64
```

0.2.5 $F(m)$ Regression Percent Error Distribution

[99]:
```
divisor_df['runtime'] = divisor_df['runtime'].replace(0, .1)
divisor_df['pct_err'] = 100*(divisor_df['regression'] - divisor_df['runtime'])/
 ↪divisor_df['runtime']
```

```python
plt.hist(divisor_df['pct_err'], bins=100)
plt.title('Divisor Regression Percent Error Histogram')
plt.xlabel('Percent Error')
plt.ylabel('Occurrences')
plt.savefig('divisor_hist.png')
```



0.2.6 $F(m)$ Regression Percent Error Sample Statistical Moments

Sample moments indicate extremely high peak with very large tails, and strong negative skew - tendency to underestimate runtimes¶

[100]:
```python
print(f'Mean = {round(np.mean(divisor_df["pct_err"]), 3)}')
print(f'Variance = {round(np.var(divisor_df["pct_err"]), 3)}')
print(f'Skew = {round(skew(divisor_df["pct_err"]), 3)}')
print(f'Kurtosis = {round(kurtosis(divisor_df["pct_err"]), 3)}')
```

```
Mean = -0.584
Variance = 35.824
Skew = -16.527
Kurtosis = 469.98
```

[101]:
```python
divisor_df['pct_err'].describe()
```

[101]:
```
count    9998.000000
mean       -0.583528
std         5.985617
min      -236.202631
25%        -1.337240
50%         1.022642
75%         1.655003
max         3.176226
Name: pct_err, dtype: float64
```