# QR-Project-20180114_final

March 24, 2018

## 1 QR Factorization via Householder Algorithm

DeJans, Adam

#### Import Libraries

```
In [1]: import pandas
        import numpy as np
        import time
        from scipy import linalg
        import scipy
        import random
        import math
```

#### HouseHolder code

```
In [2]: def householder(b):
            y = b.copy()
            y[0] = y[0] + np.sign(y[0])*np.linalg.norm(y)
            v = y / np.linalg.norm(y)
            return v

        def qr(a):
            A = a.copy()
            m, n = A.shape # Dimenstions of matrix A
            v = [0]*n
            # Check for over/under determined matrix
            if m == n: # square matrix
                r = n-1 # we don't need to manipulate the 1x1 matrix
            elif m > n: # overdetermined
                r = n
            else: # underdetermined; note this won't create an upper triangle matrix
                r = m-1
            for i in range(r):
                v[i] = householder(A[i:, i])
                vk = np.array(v[i])
                #print vk
```

```
            A[i:,i:] = A[i:,i:] - 2*np.matmul(np.matmul(vk.reshape(m-i,1),vk.reshape(1,m-i))
            #A[i:,i:] *= -1
        return A, v
```

**Testing Correctness w/ small but clear example**

```
In [3]: #a = np.random.rand(m,n)*10**random.randint(0,10)

        a = np.array(((
            (-1.2, -1.2, 1.2),
            (1, 3, 3),
            (-1, -1, 5),
            (1, 3, 7),
            (2, 36, 71),
        )))

        print 'matrix a:\n', a, '\n'

        m,n = a.shape

        r, tau = qr(a)

        q2, r2 = np.linalg.qr(a)


        print 'my r:\n', r[0:n,:].round(12), '\n'
        print 'numpy r:\n', r2.round(12), '\n'

        sR = np.subtract(r[0:n,:], r2)
        difR = np.linalg.norm(sR)

        print '||r-r2|| = ', difR
```

```
matrix a:
[[ -1.2  -1.2   1.2]
 [  1.    3.    3. ]
 [ -1.   -1.    5. ]
 [  1.    3.    7. ]
 [  2.   36.   71. ]]

my r:
[[  2.90516781  27.68858988  50.10381828]
 [  0.          23.44743036  50.84804202]
 [  0.           0.          -5.43360069]]

numpy r:
[[  2.90516781  27.68858988  50.10381828]
 [  0.          23.44743036  50.84804202]
```

```
 [  0.           0.          -5.43360069]]

||r-r2|| =  1.64732556405e-14
```

### 1.0.1 Generating random $m \times n$ matrices to test correctness

**Take the norm of the upper triangular matrix given from numpy package with my householder's algorithm and find the largest of these values.**

```
In [4]: seed = 5
        largest = 0
        for m in range(2,101):
            for n in range(2,101):
                a = np.random.rand(m,n)*10**random.randint(0,10)
                #print a
                r, tau = qr(a)
                q2, r2 = np.linalg.qr(a)
                sR = np.subtract(r[0:n,:], r2)
                difR = np.linalg.norm(sR)
                if difR > 0:
                    if difR > largest:
                        largest = difR
        print 'Largest: ', largest
        print 'Complete'
```

```
Largest:  0.00116852341016
Complete
```

### 1.0.2 Runtime Analysis

**This section was ran multiple times** The data was collected and put into csv files, which were then merged together and used to obtain the blocks that follow after this one.

```
In [ ]: timeTrial=[(0,0,0)]

        avgtime = 0
        m = 1400
        for j in range(1000):
            #m = int(math.ceil(m*1.4))
            m += 1000
            print m
            print avgtime
            for n in range(2000,m+1,100):
                timekeeper = 0
                for i in range(3):
                    a = np.random.rand(m,n)*10**random.randint(0,10)
                    now = time.time()
```

```
                q, r = qr(a)
                totTime = time.time()-now
                timekeeper += totTime
            avgtime = timekeeper / 3.0
            timeTrial.append((m,n,avgtime))
            timeTrialFrame = pandas.DataFrame(timeTrial)
            timeTrialFrame.to_csv('newData11', sep=',')

    timeTrialFrame = pandas.DataFrame(timeTrial)
    timeTrialFrame
    #timeTrialFrame.to_csv('updatedData', sep=',')
```

```
In [5]: timeTrialFrame = pandas.read_csv('mergedTimes2_edited.csv')
```

```
In [6]: timeTrialFrame.columns = ['m','n','time']
```

### 1.0.3 timeTrialFrame

```
In [7]: %matplotlib inline
        data = timeTrialFrame
        data.columns=['m','n','time']

        import matplotlib as plt
```

```
In [8]: data['mn2'] = data['m'] * data['n'] * data['n']# - (1/3)*data['n']**2
        data['divtime'] = data['time']/data['mn2']
        data['divtime'] = data['divtime']*10**9
```

#### View dataframe

```
In [9]: data
```

Out[9]:

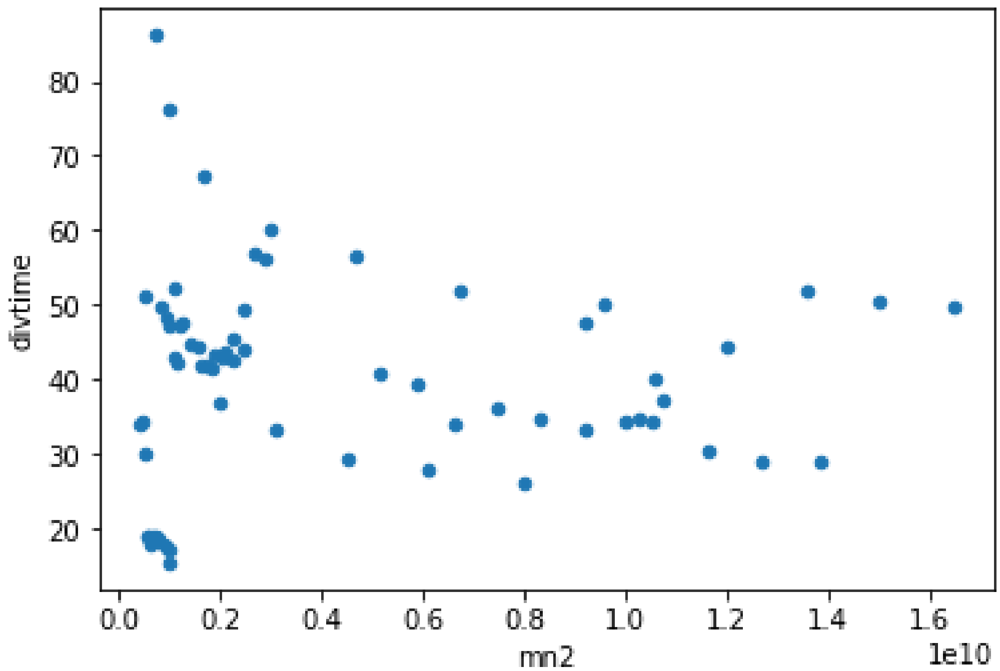
| | m | n | time | mn2 | divtime |
|---|---|---|---|---|---|
| 0 | 1000 | 750 | 10.716400 | 562500000 | 19.051378 |
| 1 | 1000 | 1000 | 15.487600 | 1000000000 | 15.487600 |
| 2 | 2000 | 500 | 25.546600 | 500000000 | 51.093200 |
| 3 | 2000 | 750 | 47.382200 | 1125000000 | 42.117511 |
| 4 | 2000 | 1000 | 73.812000 | 2000000000 | 36.906000 |
| 5 | 2000 | 1250 | 104.566200 | 3125000000 | 33.461184 |
| 6 | 2000 | 1500 | 132.624400 | 4500000000 | 29.472089 |
| 7 | 2000 | 1750 | 170.567400 | 6125000000 | 27.847739 |
| 8 | 2000 | 2000 | 208.272400 | 8000000000 | 26.034050 |
| 9 | 3000 | 500 | 64.625800 | 750000000 | 86.167733 |
| 10 | 3000 | 750 | 113.585000 | 1687500000 | 67.309630 |
| 11 | 3000 | 1000 | 180.580600 | 3000000000 | 60.193533 |
| 12 | 3000 | 1250 | 264.800800 | 4687500000 | 56.490837 |
| 13 | 3000 | 1500 | 349.739000 | 6750000000 | 51.813185 |
| 14 | 3000 | 1750 | 437.854000 | 9187500000 | 47.657578 |
| 15 | 3000 | 2000 | 532.013200 | 12000000000 | 44.334433 |

| | | | | | |
|---|---|---|---|---|---|
| 16 | 2500 | 2000 | 344.660200 | 10000000000 | 34.466020 |
| 17 | 2500 | 2025 | 354.843600 | 10251562500 | 34.613611 |
| 18 | 2500 | 2050 | 361.563000 | 10506250000 | 34.414087 |
| 19 | 2500 | 2075 | 402.847800 | 10764062500 | 37.425256 |
| 20 | 1000 | 650 | 14.457600 | 422500000 | 34.219171 |
| 21 | 1000 | 681 | 16.015200 | 463761000 | 34.533305 |
| 22 | 1000 | 712 | 15.271000 | 506944000 | 30.123643 |
| 23 | 2400 | 2100 | 425.181667 | 10584000000 | 40.172115 |
| 24 | 2400 | 2200 | 353.401333 | 11616000000 | 30.423669 |
| 25 | 2400 | 2000 | 480.467000 | 9600000000 | 50.048646 |
| 26 | 1000 | 743 | 10.480400 | 552049000 | 18.984547 |
| 27 | 1000 | 774 | 10.885400 | 599076000 | 18.170316 |
| 28 | 1000 | 805 | 11.660600 | 648025000 | 17.994059 |
| 29 | 1000 | 836 | 13.385800 | 698896000 | 19.152778 |
| .. | ... | ... | ... | ... | ... |
| 38 | 2000 | 743 | 47.322400 | 1104098000 | 42.860688 |
| 39 | 2000 | 774 | 56.508000 | 1198152000 | 47.162630 |
| 40 | 2000 | 700 | 74.810400 | 980000000 | 76.337143 |
| 41 | 2000 | 746 | 58.267600 | 1113032000 | 52.350337 |
| 42 | 2000 | 792 | 59.977200 | 1254528000 | 47.808578 |
| 43 | 2000 | 838 | 63.106800 | 1404488000 | 44.932246 |
| 44 | 2000 | 884 | 69.381600 | 1562912000 | 44.392519 |
| 45 | 2000 | 930 | 72.395800 | 1729800000 | 41.852122 |
| 46 | 2000 | 976 | 82.858200 | 1905152000 | 43.491648 |
| 47 | 2000 | 1022 | 91.643800 | 2088968000 | 43.870370 |
| 48 | 2000 | 1068 | 97.100000 | 2281248000 | 42.564421 |
| 49 | 2000 | 1114 | 109.763200 | 2481992000 | 44.223833 |
| 50 | 2000 | 1160 | 152.930000 | 2691200000 | 56.825951 |
| 51 | 2000 | 1206 | 163.498800 | 2908872000 | 56.206942 |
| 52 | 2000 | 900 | 68.147200 | 1620000000 | 42.066173 |
| 53 | 2000 | 953 | 75.241600 | 1816418000 | 41.423065 |
| 54 | 2000 | 1006 | 87.142400 | 2024072000 | 43.053014 |
| 55 | 2000 | 1059 | 102.156200 | 2242962000 | 45.545221 |
| 56 | 2000 | 1112 | 122.391600 | 2473088000 | 49.489383 |
| 57 | 2300 | 1500 | 211.474000 | 5175000000 | 40.864541 |
| 58 | 2300 | 1600 | 232.178600 | 5888000000 | 39.432507 |
| 59 | 2300 | 1700 | 225.706400 | 6647000000 | 33.956131 |
| 60 | 2300 | 1800 | 270.789400 | 7452000000 | 36.337815 |
| 61 | 2300 | 1900 | 287.194600 | 8303000000 | 34.589257 |
| 62 | 2300 | 2000 | 305.880600 | 9200000000 | 33.247891 |
| 63 | 2400 | 2300 | 368.011333 | 12696000000 | 28.986400 |
| 64 | 2400 | 2400 | 403.468000 | 13824000000 | 29.186053 |
| 65 | 3400 | 2000 | 708.008667 | 13600000000 | 52.059461 |
| 66 | 3400 | 2100 | 758.158000 | 14994000000 | 50.564092 |
| 67 | 3400 | 2200 | 816.510667 | 16456000000 | 49.617809 |

[68 rows x 5 columns]

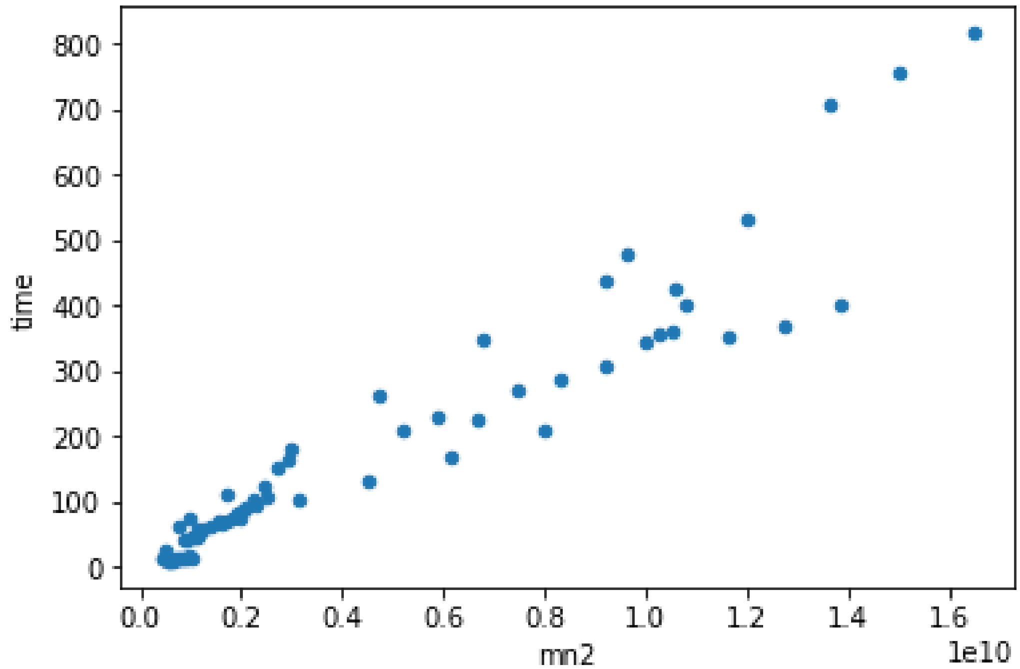
**Plot dataframe** *mn^2 by divtime*

```
In [10]: data.plot.scatter(x='mn2', y='divtime')

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xc34f240>
```




**Plot dataframe** *mn^2 by time*

```
In [11]: data['mn2'] = data['m'] * data['n'] * data['n'] #- (1/3)*data['n']**2
         data['divtime'] = data['time']/data['mn2']
         data['divtime'] = data['divtime']#*10**8
         data.plot.scatter(x='mn2', y='time')

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0xc5c8668>
```

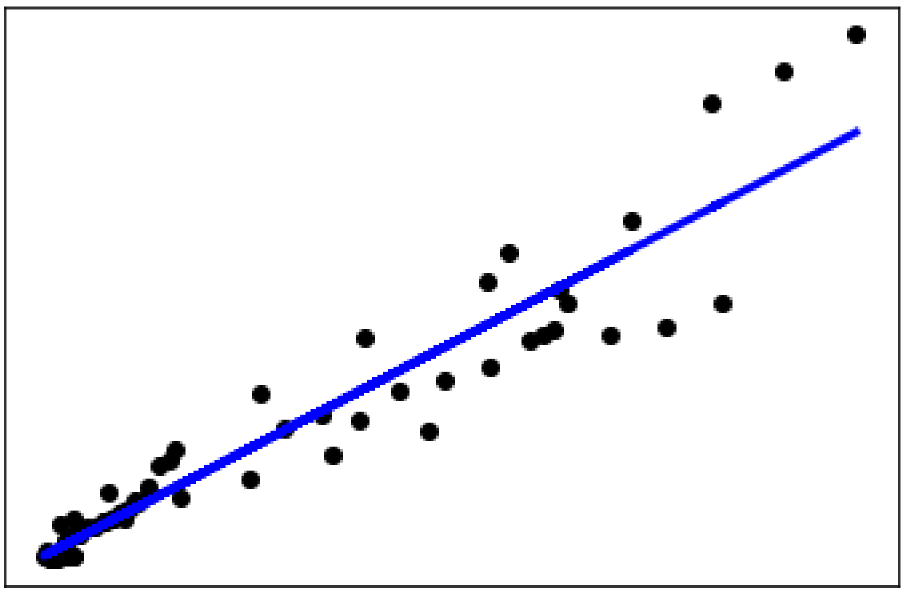



**Plot dataframe** Sketching line for *mn^2 by time*

```
In [12]: from sklearn import datasets, linear_model
         from sklearn.metrics import mean_squared_error, r2_score
         import matplotlib.pyplot as plt #this allows us to use 'plt.scatter'
         x = data['mn2']
         y = data['time']

         c = len(data.index)

         x = x.reshape(c, 1)
         y = y.reshape(c, 1)
         regr = linear_model.LinearRegression()
         regr.fit(x, y)

         # plot it as in the example at http://scikit-learn.org/
         plt.scatter(x, y,  color='black')
         plt.plot(x, regr.predict(x), color='blue', linewidth=3)
         plt.xticks(())
         plt.yticks(())
         plt.show()

         print('Coefficients: \n', regr.coef_)
         # The mean squared error
```

```
    print("Mean squared error: %.2f"
          % mean_squared_error(x, y))
    # Explained variance score: 1 is perfect prediction
    print('Variance score: %.2f' % r2_score(x, y))
```

```
C:\Users\adamd\Anaconda2\lib\site-packages\ipykernel_launcher.py:9: FutureWarning: reshape is de
  if __name__ == '__main__':
C:\Users\adamd\Anaconda2\lib\site-packages\ipykernel_launcher.py:10: FutureWarning: reshape is d
  # Remove the CWD from sys.path while we load stuff.
```



```
('Coefficients: \n', array([[  4.03927657e-08]]))
Mean squared error: 388742762289467064 32.00
Variance score: -0.98
```

*Note: there was more testing that was done, but it was not saved to the notebook. This notebook covers the majority.*

In [ ]: