40.004 Statistics 2018: Problem set 3

due Mon, 9 April, 2018 at 11:59 pm. Submit on e-dimension.

Done by Adam Ilyas 1002010

In [1]:
```
from scipy import stats
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import math
%matplotlib inline
```

In [2]:
```
def test(p_value, alpha):
    if p_value>=alpha:
        print("Since {} >= {} \nwe do not reject our hypothesis".format(p_
value, alpha))
    else:
        print("Since {} < {} \nwe reject our hypothesis".format(p_value, a
lpha))
```

Question 1

People at high risk of sudden cardiac death can be identified using the change in a signal averaged electrocardiogram (ECG) before and after the prescribed activities. The current method is 80% accurate. The method was modified, hoping to improve its accuracy. The new method is tested on 50 people and gave correct results on 46 patients. Is this convincing evidence that the new method is accurate

a) Set up the hypotheses to test that the accuracy of the new method is better than that of the current method.

H_0: Success rate of new method $\leq$ 0.8

H_A: Success rate of new method > 0.8

b) Perform a test of the hypotheses at level α = 0.05. What do you conclude about the accuracy of the method?

For method 1: Each of the result have a 80% probability of being correct. Thus 40 out of 50 trials were correct (successful)

Let X be the number of successful trials:

X ~ Normal (40, 0.80)

Assuming that H_0 is true, the probability of a result as extreme (or more) than 46 out 50 successful trials is as follows:

$P(X \geq 46) = P(Z \geq \frac{46-40}{\sqrt{8/50}})$

In [3]:
```
p_value = 1 - stats.norm.cdf((46-40)/((8/50)**0.5))
alpha = 0.05
print('P-value: {} \n'.format(p_value))
if p_value>=alpha:
    print("Since {} >= {} \nwe do not reject our hypothesis".format(p_valu
e, alpha))
else:
    print("Since {} < {} \nwe reject our hypothesis".format(p_value, alpha
))
```

```
P-value: 0.0

Since 0.0 < 0.05
we reject our hypothesis
```

c) If the new method actually has 90% accuracy, what power does a sample of 50 have to demonstrate that the new method is better, using a 0.05-level test?

Power = 1 - β

P(X > 40 | μ = 0.9):

d) How many patients should be tested in order for this power to be at least 0.75?

In [ ]:
```

```

Question 2

We have looked at the distribution of the digits of π in base 10, and the result is consistent with the null hypothesis that the digits of π are random. However, it is plausible for a number to appear random in one base but not in another. The spreadsheet shows the distribution of the digits of π in base 16.

In [4]:
```
df = pd.read_excel('stats2018_PS3.xlsx', sheetname=0)

df.head()
```

Out[4]:

	Hex digit	Occurrences
0	0	62499881108
1	1	62500212206
2	2	62499924780
3	3	62500188844

4	4	62499807368

In [5]:
```
df['Expected'] = [sum(df['Occurrences'])/16 for i in range(16)]
df['test statistic'] = (df['Occurrences'] - df['Expected'])**2/df['Expected']

df
```

Out[5]:

	Hex digit	Occurrences	Expected	test statistic
0	0	62499881108	6.250000e+10	0.226165
1	1	62500212206	6.250000e+10	0.720502
2	2	62499924780	6.250000e+10	0.090529
3	3	62500188844	6.250000e+10	0.570593
4	4	62499807368	6.250000e+10	0.593713
5	5	62500007205	6.250000e+10	0.000831
6	6	62499925426	6.250000e+10	0.088981
7	7	62499878794	6.250000e+10	0.235054
8	8	62500216752	6.250000e+10	0.751703
9	9	62500120671	6.250000e+10	0.232984
10	A	62500266095	6.250000e+10	1.132905
11	B	62499955595	6.250000e+10	0.031549
12	C	62500188610	6.250000e+10	0.569180
13	D	62499613666	6.250000e+10	2.388063
14	E	62499875079	6.250000e+10	0.249684
15	F	62499937801	6.250000e+10	0.061899

a) Perform a chi-squared test on the null hypothesis, using α = 0.05.

H_0: Each digit has a probability of 1/16 of occuring

H_A: At least one of digit has a probability != 1/16

In [6]:
```
statistic = sum(df['test statistic'])
n = 16
p_value = 1 - stats.chi2.cdf(statistic, df=n-1)
print('Chisquare statistic: {} \n'.format(p_value))
test(p_value, alpha = 0.05)
```

```
Chisquare statistic: 0.9259949835349345

Since 0.9259949835349345 >= 0.05
we do not reject our hypothesis
```

b) Give an example of a number whose base 10 digits are clearly not random, yet the chi-squared test fails to reject the null hypothesis

H_0: The data are consistent with a specified distribution.

H_A: The data are not consistent with a specified distribution.

$0.\overline{1234567890}$

The value above is clearly not random but using the method above, each digit will have equal distribution (except 0) and will fail to reject the null hypothesis

Question 3

While trying to find the least square regression line for some data points (xi, yi), a statistician, giddy from the recent Putin's landslide victory, used the points (yi , xi) instead.

a) Does the correlation coefficient of the resulting regression line agree with the correct r?

In [7]:
```
x = np.random.uniform(0,1,101)
y = np.random.uniform(0,1,101)
```

In [8]:
```
sample_covariance = np.cov(x,y, ddof=1)
std_x = np.std(x, ddof=1)
std_y = np.std(y, ddof=1)
```

In [9]:
```
np.corrcoef(x,y)[0,1]
```

Out[9]:
```
0.028742902442123696
```

In [10]:
```
np.corrcoef(x,y)[1,0]
```

Out[10]:
```
0.028742902442123696
```

correlation coefficient does not change

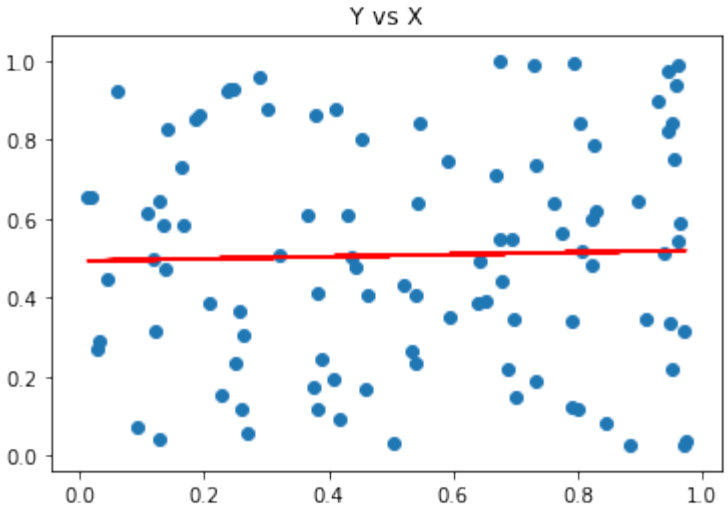
b) What about the slope of the resulting regression line: is it the same as the correct $\hat{\beta}_1$, or is it $\hat{\beta}_1$ flipped around the line y = x, or does something else happen?

In [11]:
```
result = stats.linregress(x,y)
print('Correlation Coefficient: {} \nSlope: {}\nIntercept: {}'.format(resu
lt.rvalue, result.slope, result.intercept))
plt.title('Y vs X')
plt.scatter(x,y)
plt.plot(x, (x*result.slope+result.intercept), 'r')
```

```
Correlation Coefficient: 0.028742902442123696
Slope: 0.027358912721186228
```

```
Intercept: 0.49243189223904593
```

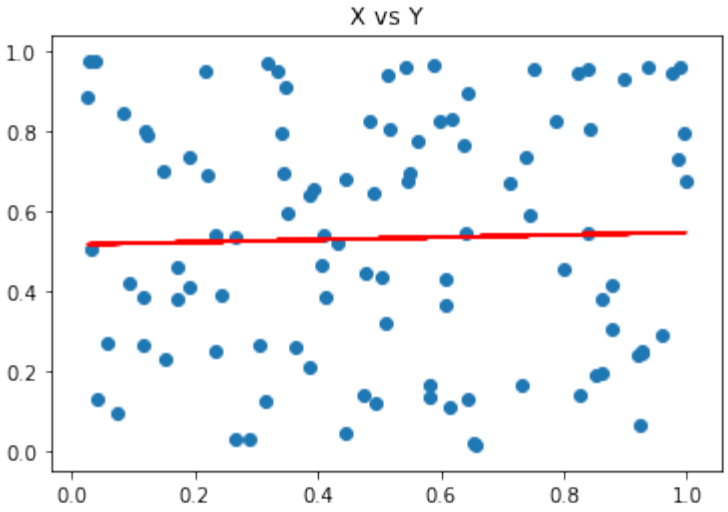
Out[11]: [<matplotlib.lines.Line2D at 0x1c2e822f9b0>]



In [12]:
```
result = stats.linregress(y,x)
print('Correlation Coefficient: {} \nSlope: {}\nIntercept: {}'.format(resu
lt.rvalue, result.slope, result.intercept))
plt.title('X vs Y')
plt.scatter(y,x)
plt.plot(y, (y*result.slope+result.intercept), 'r')
```

```
Correlation Coefficient: 0.028742902442123696
Slope: 0.030196903262083286
Intercept: 0.5151367390802445
```

Out[12]: [<matplotlib.lines.Line2D at 0x1c2e82fbc50>]



Question 4

Refer to the spreadsheet for some triple jump data.

In [13]:
```
df = pd.read_excel('stats2018_PS3.xlsx', sheetname=1)
df
```

Out[13]:

	year	winning distance (Olympic triple jump, women)
0	1996	15.33
1	2000	15.20
2	2004	15.30
3	2008	15.39
4	2012	14.98
5	2016	15.17

In [14]:
```
df.columns = ['year', 'distance']
df
```

Out[14]:

	year	distance
0	1996	15.33
1	2000	15.20
2	2004	15.30
3	2008	15.39
4	2012	14.98
5	2016	15.17

a) Using the formulas given in class, compute the following

The p-value of $\hat{\beta}_1$, under the hypotheses:

$H_0 : \hat{\beta}_1 = 0$

$H_1 : \hat{\beta}_1 \neq 0$

We can reject H_0 if $|r| \sqrt{n-2}\sqrt{1-r^2} > t_{n-2,1-\alpha/2}$

In [15]:
```
x = df['year']
y = df['distance']

stats.linregress(x,y)
```

Out[15]:
```
LinregressResult(slope=-0.009785714285714267, intercept=34.85847619047615,
 rvalue=-0.49946031170932303, pvalue=0.3131073676960041, stderr=0.00848688
4639510186)
```

In [16]:
```
s_xy = np.cov(x,y)[0,1]
s_x = np.std(x, ddof=1)
s_y = np.std(y, ddof=1)
```

```
r = s_xy/(s_x*s_y)
print(s_xy, s_x, s_y)
print(r)
```

```
-0.547999999999999 7.483314773547883 0.14661741597322842
-0.49946031170932315
```

In [17]:
```
n = len(x)
statistic = ( abs(r)*(n-2)**0.5)/ (1-r**2)**0.5

statistic ###################################
```

Out[17]:
```
1.1530396254189033
```

b) Using the regression line, predict the winning distance for 2020, and provide a 99% two-sided prediction interval.

In [18]:
```
x_bar = np.mean(x)
y_bar = np.mean(y)

slope = s_xy/ s_x**2
intercept = y_bar - slope * x_bar
print('Slope: {}\nIntercept: {}'.format(slope, intercept))
```

```
Slope: -0.009785714285714269
Intercept: 34.85847619047616
```

In [19]:
```
winning_distance = intercept + slope * 2020
print("Winning distance for 2020 is {}m".format(round(winning_distance,3))
)

standard_error = s_y * ( 1+1/n+ (2020-x_bar)*2/ ((n-1)* s_x) )**0.5

t = stats.t.ppf(0.995, df=n-2)
prediction_interval = [winning_distance-t*standard_error, winning_distance
+t*standard_error]

print("Prediction interval is {}".format(prediction_interval))
```

```
Winning distance for 2020 is 15.091m
Prediction interval is [14.157188956489943, 16.025477710176734]
```

c) What would be the 99% confidence interval for the winning distance for the year 2020?

In [20]:
```
standard_error = s_y * ( 1/n+ (2020-x_bar)*2/ ((n-1)* s_x) )**0.5

t = stats.t.ppf(0.995, df=n-2)

confidence_interval = [winning_distance-t*standard_error, winning_distance
+t*standard_error]

print("Prediction interval is {}".format(confidence_interval))
```

```
Prediction interval is [14.445619736081463, 15.737046930585214]
```

d) Comment on why the two intervals from parts (b) and (c) are different. What are the two different questions about the winning distance that parts (b) and (c) are asking?

Prediction of a single value of y will vary more than its expected value hence due to this added variability (as seen in the formula where the standard error has an addition +1), the prediction interval will be larger than the confidence interval

Question 5

A prime is a positive integer that has no integer factors other than 1 and itself (1 is not regarded as prime). The number of primes in any given interval of whole numbers is highly irregular. However, the proportion of primes less than or equal to any given number x (denoted p(x)) follows a regular pattern as x increases. Refer to the spreadsheet for the table that gives a number and proportion of primes for $x = 10^n$, for n = 1, . . . , 10.

a) Plot the proportion of primes, p(x), against the following:

10000/x,

$1000/\sqrt{x}$

$1/\log_{10}(x)$.

Which relationship appears the most linear

In [21]:
```
df = pd.read_excel('stats2018_PS3.xlsx', 2)

df.columns = ['x', 'Primes', 'Proportion']

def toNumber(x):
    num = x.split("^")
    return float(num[0])**int(num[1])

df['x'] = df['x'].apply(toNumber)
df.head()
```

Out[21]:

	x	Primes	Proportion
0	10.0	4	0.40000
1	100.0	25	0.25000
2	1000.0	168	0.16800
3	10000.0	1229	0.12290
4	100000.0	9592	0.09592

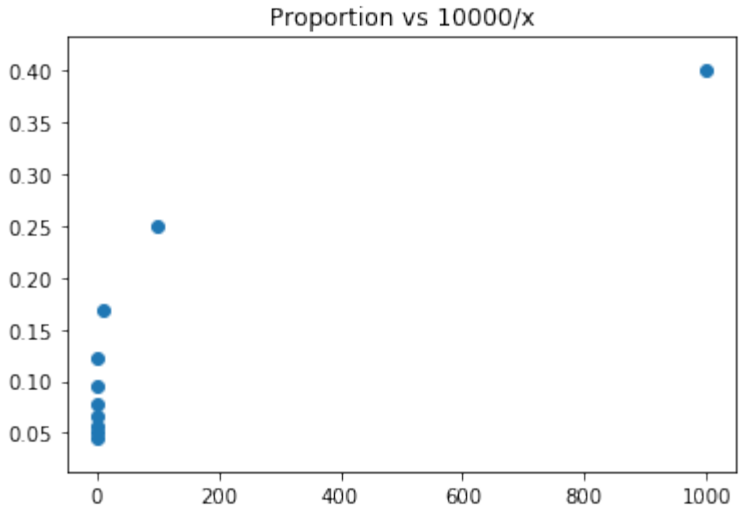
In [22]:
```
df['10000/x'] = df['x'].apply(lambda x: 10000/x)
df['1000/x^0.5'] = df['x'].apply(lambda x: 1000/x**0.5)
df['1/log10(x)'] = df['x'].apply(lambda x: 1/math.log(x,10))
```

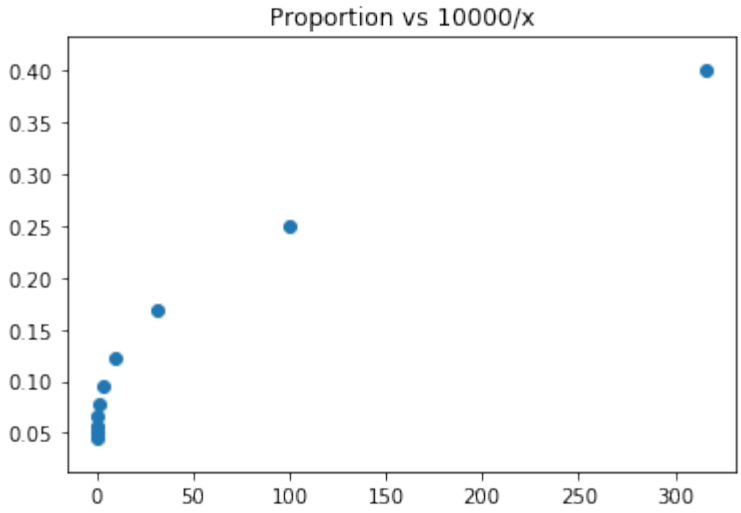
In [23]:
```
df
```

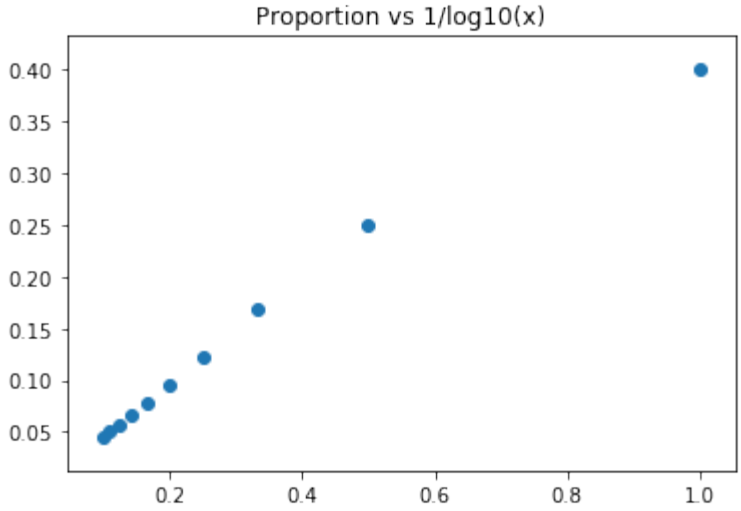
Out[23]:

	x	Primes	Proportion	10000/x	1000/x^0.5	1/log10(x)
0	1.000000e+01	4	0.400000	1000.000000	316.227766	1.000000
1	1.000000e+02	25	0.250000	100.000000	100.000000	0.500000
2	1.000000e+03	168	0.168000	10.000000	31.622777	0.333333
3	1.000000e+04	1229	0.122900	1.000000	10.000000	0.250000
4	1.000000e+05	9592	0.095920	0.100000	3.162278	0.200000
5	1.000000e+06	78498	0.078498	0.010000	1.000000	0.166667
6	1.000000e+07	664579	0.066458	0.001000	0.316228	0.142857
7	1.000000e+08	5761455	0.057615	0.000100	0.100000	0.125000
8	1.000000e+09	50847534	0.050848	0.000010	0.031623	0.111111
9	1.000000e+10	455052512	0.045505	0.000001	0.010000	0.100000

In [24]:
```
plt.scatter(df['10000/x'], df['Proportion'])
plt.title('Proportion vs 10000/x')
plt.show()

plt.scatter(df['1000/x^0.5'], df['Proportion'])
plt.title('Proportion vs 10000/x')
plt.show()

plt.scatter(df['1/log10(x)'], df['Proportion'])
plt.title('Proportion vs 1/log10(x)')
plt.show()
```







b) Produce a regression line for the linearized data, and hence find an equation relating p(x) and x. Test whether the observed $\hat{\beta}_1$ matches the theoretical $\hat{\beta}_1 = \log_{10}(e) = 0.4343$.

In [25]:
```
x = df['1/log10(x)']
y = df['Proportion']
n = len(x)

results = stats.linregress(x,y)
print('Slope: {}\nIntercept: {}'.format(results.slope, results.intercept))
```

```
Slope: 0.4041915868937658
Intercept: 0.015187967346710907
```

H_0: $\hat{\beta}_1 = \log_{10}(e)$

H_A: $\hat{\beta}_1 \neq \log_{10}(e)$

In [26]:
```
se = ( np.std(y,ddof=2) / np.std(x, ddof=1)) / (n-1)**0.5
t = stats.t.ppf(0.975 , df=n-2)
confidence_interval = [results.slope-t*se, results.slope+t*se]
```

```
confidence_interval ###################################

if confidence_interval[0]<=results.slope<=confidence_interval[1]:
    print('Since {} lies with {}, we do not reject null hypothesis'.format
(results.slope, confidence_interval))
else:
    print('Since {} does not lies with {}, we reject null hyppthesis'.form
at(results.slope, confidence_interval))
```

```
Since 0.4041915868937658 lies with [0.07156733167242496, 0.736815842115106
5], we do not reject null hypothesis
```

c) Verify whether $\hat{\beta}_0$ is significantly different from zero.

And thus explain how this translates into the Prime Number Theorem: for large x, p(x) ≈ 1/ ln x

In [27]:
```
# for beta 0
se = ( np.std(y,ddof=2) / np.std(x, ddof=1)) * ((np.std(x,ddof=2)**2 / n)
+ (np.mean(x)**2 / (n-1))) **0.5
t = stats.t.ppf(0.975 , df=n-2)
confidence_interval = [results.intercept-t*se, results.intercept+t*se]

if confidence_interval[0]<=0<=confidence_interval[1]:
    print('Since {} lies with {}, we do not reject null hypothosis'.format
(0, confidence_interval))
else:
    print('Since {} does not lies with {}, we reject null hyppthesis'.form
at(0, confidence_interval))
```

```
Since 0 lies with [-0.11936062060305708, 0.1497365552964789], we do not re
ject null hypothosis
```

Question 6.

The Salk polio vaccine trial was the most elaborate program of its kind ever, involving millions of participants. Some of its results are given in the spreadsheet: out of the 200,000+ people given a placebo, 142 developed polio, while out of a similar number of people given the vaccine, 57 did. We would like to test (and hopefully, reject) the null hypothesis that the vaccine is ineffective. To do so, we could rephrase H_0 as: the incidence of polio is independent of which group one is in. Pick a reasonable α and perform a chi-squared test for independence to test this.

In [28]:
```
df = pd.read_excel('stats2018_PS3.xlsx', 3)
df
```

Out[28]:

	Vaccine	Placebo	total
Polio	57	142	199
No polio	200688	201087	401775
total	200745	201229	401974

H_0: (Vaccine is ineffective) incidence of polio is independent of which group (Vaccine/ Placebo)

H_A: incidence of polio is dependant of which group (Vaccine/ Placebo)

In [29]:
```
observed = np.array(df.iloc[[0,1],[0,1]])
observed
```

Out[29]:
```
array([[    57,    142],
       [200688, 201087]], dtype=int64)
```

In [30]:
```
polio_row_total = 199
no_polio_row_total = 401775

vaccine_col_total = 200745
placebo_col_total = 201229

overall_total = 401974
```

In [31]:
```
expected = np.array([[polio_row_total*vaccine_col_total/ overall_total,
                              polio_row_total*placebo_col_total/ overall_to
tal],
                             [no_polio_row_total*vaccine_col_total/ overall
_total,
                              no_polio_row_total*placebo_col_total/ overall
_total]])

expected
```

Out[31]:
```
array([[9.93801962e+01, 9.96198038e+01],
       [2.00645620e+05, 2.01129380e+05]])
```

In [32]:
```
n = len(observed.flatten()) # no of observations
df = (observed.shape[0]-1)*(observed.shape[1]-1)
statistic = sum([(expected.flatten()[i]-observed.flatten()[i])**2/ expecte
d.flatten()[i] for i in range(4)])

p_value = 1 -stats.chi2.cdf(statistic, df=df)

p_value
```

Out[32]:
```
1.8552664959869958e-09
```

Since the P value is extremely small, we can reject the null hypothesis that the vaccine is ineffective!

Question 7

In [ ]:
```
```

Question 8

In [ ]:
```
```

Question 9

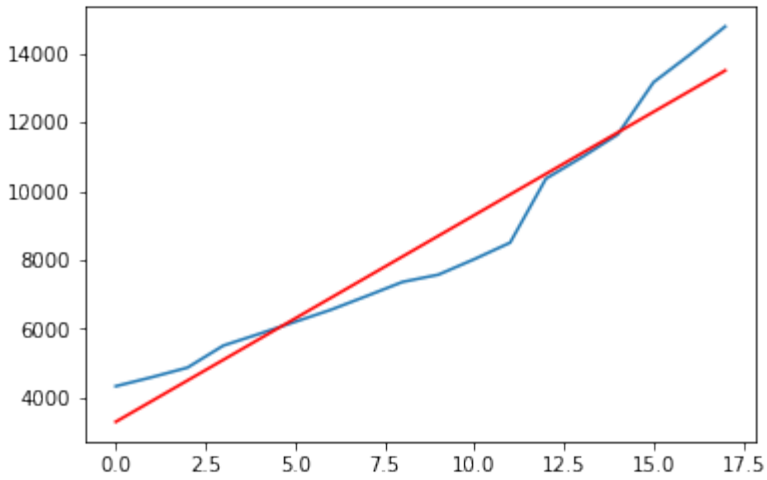
In [40]:
```
df = pd.read_excel('stats2018_PS3.xlsx', 4)
df.columns = ['Age', 'Cost']
df.head()
```

Out[40]:

	Age	Cost
0	0	4330
1	1	4590
2	2	4870
3	3	5510
4	4	5850

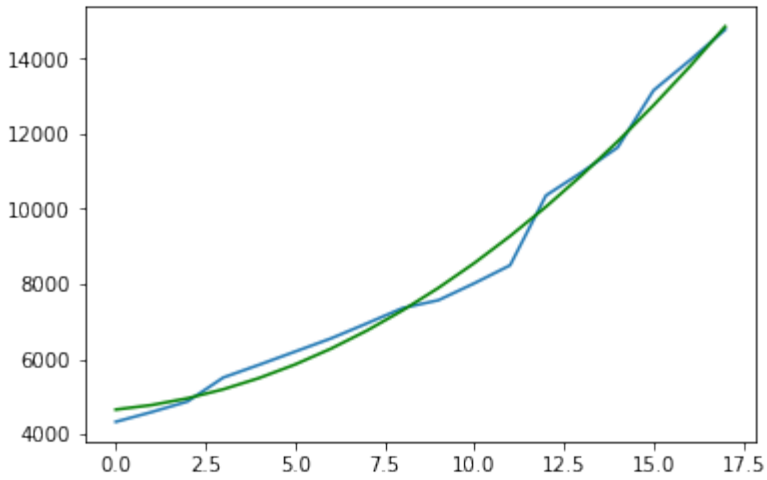
In [41]:
```
x = df['Age']
y = df['Cost']

plt.plot(x, y)
results1 = np.polyfit(x,y, deg=1)
plt.plot(x, (x*results1[0]+results1[1]), 'r')
```

Out[41]:
```
[<matplotlib.lines.Line2D at 0x1c2e9724668>]
```



In [44]:
```
plt.plot(x, y)
results2 = np.polyfit(x,y, deg=2)
plt.plot(x, (results2[0]*x**2+results2[1]*x+results2[2]), 'g')
results2
```

Out[44]:
```
array([  30.01934985,   89.99097007, 4656.49122807])
```



b) Fit a straight line $y = \beta_0 + \beta_1 t$, where y is the annual cost of raising a child and t is the child's age. What is r^2 and r_{2adj} ?

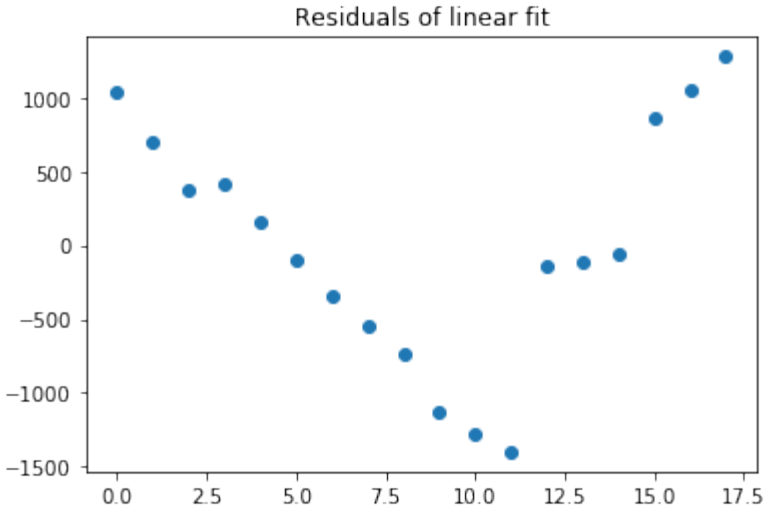
In [36]:
```
n = len(x)

s_x = np.std(x,ddof=1)
s_y = np.std(y,ddof=1)
s_xy = np.cov(x,y)[0,1]
r2 = (s_xy/(s_x*s_y))**2
r2_adj = 1 - (n-1)/(n-1-1)*(1-r2)

print('r squared: {}\nr squared adjusted: {}'.format(r2, r2_adj))
```

```
r squared: 0.9395178093915884
r squared adjusted: 0.9357376724785627
```

c) Plot the residuals against t. What does the residual plot indicate about the linear fit?

In [52]:
```
x_bar = np.mean(x) # age
y_bar = np.mean(y) # cost

results1 = np.polyfit(x,y, deg=1)
residuals = [y[i] - (x[i]*results1[0]+results1[1]) for i in range(len(y))]

plt.scatter(x, residuals)
plt.title('Residuals of linear fit')
```

Out[52]:
```
<matplotlib.text.Text at 0x1c2e9958438>
```



d) Now fit a quadratic $y = \beta_0 + \beta_1 t + \beta_2 t^2$. Compute the r^2 and r_{2adj} .

How much additional variation in y is accounted for by the quadratic term? Comment on how r^2 and r_{2adj} change as compared to part (b)

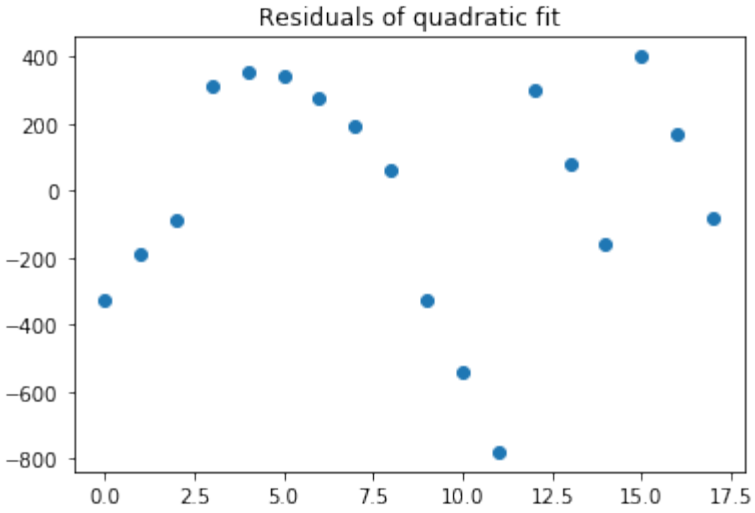
In [48]:
```
results2 = np.polyfit(x,y, deg=2)
y2 = results2[0]*x**2+results2[1]*x+results2[2]

s_x = np.std(x,ddof=1)
s_y2 = np.std(y2,ddof=1)
s_xy_new = np.cov(x,y2)[0,1]
r2_new = (s_xy_new/(s_x*s_y2))**2
r2_adj_new = 1 - (n-1)/(n-1-1)*(1-r2_new)

print('r squared: {}\nr squared adjusted: {}'.format(r2_new, r2_adj_new))
```

```
r squared: 0.9493563456652855
r squared adjusted: 0.9461911172693658
```

e) Plot the residuals of the quadratic fit against t. How good is the quadratic fit? Are there any outliers?

In [51]:
```
residuals2 = [y[i] - y2[i] for i in range(len(y))]

plt.scatter(x, residuals2)
plt.title('Residuals of quadratic fit')
```

Out[51]:
```
<matplotlib.text.Text at 0x1c2e98c1160>
```



In [55]:
```
df['z'] = df['Age'].apply(lambda x: 0 if x<12 else 1)
df['tz'] = df['Age']*df['z']
df
```

Out[55]:

	Age	Cost	z	tz
0	0	4330	0	0
1	1	4590	0	0
2	2	4870	0	0
3	3	5510	0	0
4	4	5850	0	0
5	5	6200	0	0
6	6	6550	0	0
7	7	6950	0	0
8	8	7360	0	0
9	9	7570	0	0
10	10	8020	0	0
11	11	8500	0	0
12	12	10360	1	12
13	13	10980	1	13
14	14	11640	1	14
15	15	13160	1	15
16	16	13950	1	16
17	17	14780	1	17

In [57]:
```
from sklearn import linear_model
df.columns
```

Out[57]:
```
Index(['Age', 'Cost', 'z', 'tz'], dtype='object')
```

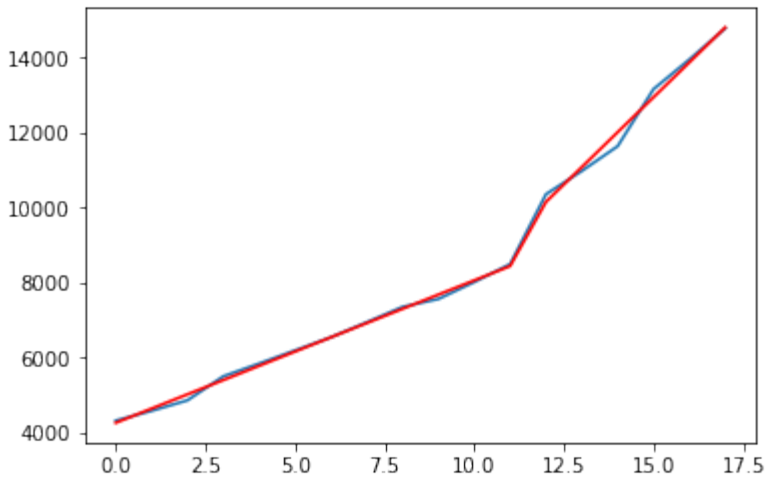
In [ ]:
```
## Create linear regression object
regr = linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(df[['Age', 'z', 'tz']], df['Cost'])
```

In [63]:
```
[b1, b2, b3] = regr.coef_
b0 = regr.intercept_
print("b0: {} \nb1: {}\nb2: {}\nb3: {}".format(b0, b1, b2, b3))
```

```
b0: 4271.025641025651
b1: 379.51048951048864
b2: -5269.406593406596
b3: 549.9180819180817
```

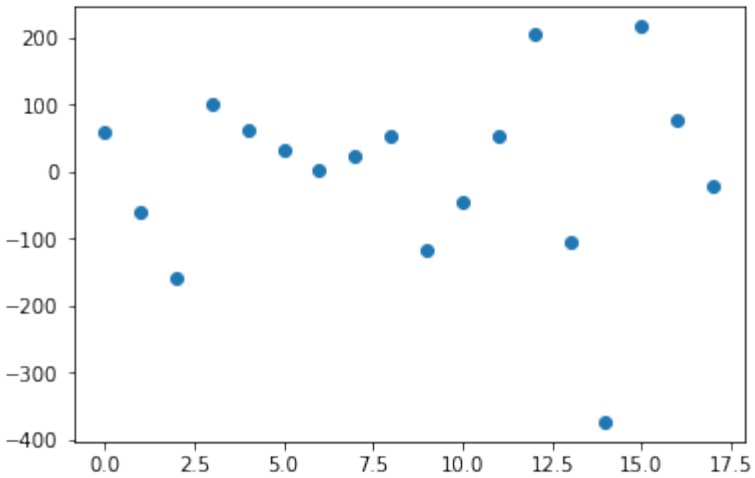
In [66]:
```
y_pred = regr.predict(df[['Age', 'z', 'tz']])
plt.plot(x, y)
plt.plot(x, y_pred, 'r')
```

Out[66]:
```
[<matplotlib.lines.Line2D at 0x1c2e9f620f0>]
```



In [68]:
```
residuals = [y[i] - y_pred[i] for i in range(len(y))]

plt.scatter(x, residuals)
```

Out[68]:
```
<matplotlib.collections.PathCollection at 0x1c2ea210eb8>
```



In [ ]: